

mitsubishi

PROGRAMMABLE CONTROLLER

MELSEC-A

User's Manual

**Intelligent communication module
type AD51-S3**

REVISIONS

※The manual number is given on the bottom left of the back cover.

| Print Date | *Manual Number | Revision |
|------------|-----------------|---------------|
| Jan., 1989 | IB (NA) 66189-A | First edition |

INTRODUCTION

Thank you for choosing the Mitsubishi MELSEC-A Series of General Purpose Programmable Controllers. Please read this manual carefully so that the equipment is used to its optimum. A copy of this manual should be forwarded to the end User.

| | | |
|------|--|------|
| 1 | INTRODUCTION | 1-1 |
| 2 | SYSTEM CONFIGURATION | 2-1 |
| 3 | SPECIFICATIONS | 3-1 |
| 4 | PRE-START UP PROCEDURES | 4-1 |
| 5 | WIRING | 5-1 |
| 6 | AD51 PROGRAMMING NOTES | 6-1 |
| 7 | COMMUNICATION WITH PROGRAMMABLE CONTROLLER CPU | 7-1 |
| 8 | TROUBLESHOOTING | 8-1 |
| 9 | MAINTENANCE | 9-1 |
| | APPENDICES | |
| A-1 | Appendix A: System Configuration | A-1 |
| A-2 | Appendix B: System Specifications | A-2 |
| A-3 | Appendix C: Pre-Start Up Procedures | A-3 |
| A-4 | Appendix D: Wiring Diagrams | A-4 |
| A-5 | Appendix E: AD51 Programming Notes | A-5 |
| A-6 | Appendix F: Communication with Programmable Controller CPU | A-6 |
| A-7 | Appendix G: Troubleshooting | A-7 |
| A-8 | Appendix H: Maintenance | A-8 |
| A-9 | Appendix I: Glossary | A-9 |
| A-10 | Appendix J: Index | A-10 |

CONTENTS

| | |
|--|------------|
| 1. INTRODUCTION | 1-1 ~ 1-6 |
| 1.1 Notes on Character Sets | 1-6 |
| 2. SYSTEM CONFIGURATION | 2-1 ~ 2-6 |
| 2.1 Overall Configuration | 2-1 |
| 2.2 Applicable A Series Systems | 2-4 |
| 2.3 Peripheral Equipment | 2-5 |
| 3. SPECIFICATIONS | 3-1 ~ 3-26 |
| 3.1 Performance Specifications | 3-1 |
| 3.2 Instruction Set | 3-2 |
| 3.2.1 GPC-BASIC commands | 3-2 |
| 3.2.2 System subroutines | 3-6 |
| 3.3 Software Configuration | 3-9 |
| 3.4 Memory Map | 3-10 |
| 3.4.1 Memory configuration | 3-10 |
| 3.4.2 Memory map with ROM loaded | 3-11 |
| 3.4.3 Storing system data on ROM | 3-13 |
| 3.5 Interface Specifications | 3-14 |
| 3.5.1 RS-422 connector specifications (CH1) | 3-14 |
| 3.5.2 RS-422 terminal block specifications (CH2) | 3-15 |
| 3.5.3 RS-232C connector performance specifications (CH3 and 4) | 3-16 |
| 3.6 I/O Interface with Programmable Controller CPU | 3-17 |
| 3.7 Buffer Memory | 3-19 |
| 3.8 Communication between AD51 and PC CPU | 3-20 |
| 3.8.1 Transmission time in MELSECNET | 3-22 |
| 3.9 AD51 Communication | 3-23 |
| 3.10 Communication Control | 3-25 |
| 3.10.1 DTR control | 3-26 |
| 3.10.2 Xon/Xoff control | 3-26 |
| 4. PRE-START UP PROCEDURES | 4-1 ~ 4-11 |
| 4.1 General Procedure | 4-1 |
| 4.2 Nomenclature | 4-2 |
| 4.3 Hardware Settings | 4-5 |
| 4.3.1 Memory protect range | 4-5 |
| 4.3.2 Console channel | 4-7 |
| 4.3.3 Terminal resistor | 4-7 |
| 4.3.4 Setting system data transfer | 4-8 |
| 4.3.5 ROM installation | 4-9 |
| 4.3.6 Loading the battery | 4-11 |

| | |
|--|------------|
| 5. WIRING | 5-1 ~ 5-3 |
| 5.1 Wiring Instructions | 5-1 |
| 5.2 RS-232C Connection | 5-1 |
| 5.3 RS-422 Connection | 5-2 |
| 6. AD51 PROGRAMMING NOTES | 6-1 ~ 6-12 |
| 6.1 BASIC Program Address Data | 6-1 |
| 6.2 Start Conditions | 6-4 |
| 6.2.1 Program runs once after power on | 6-4 |
| 6.2.2 Program runs continuously after power on | 6-4 |
| 6.2.3 Program runs after an interrupt signal from the PC CPU | 6-5 |
| 6.2.4 Program runs at preset intervals in real time | 6-5 |
| 6.3 Notes on the Use of BASIC Commands | 6-6 |
| 6.3.1 Key input commands | 6-6 |
| 6.3.2 Printing commands | 6-6 |
| 6.3.3 CRT display commands | 6-8 |
| 6.3.4 OPEN and CLOSE commands | 6-9 |
| 6.3.5 Z commands | 6-9 |
| 6.4 Transmission Commands to External Device | 6-10 |
| 6.4.1 PRINT command | 6-10 |
| 6.4.2 LPRINT command | 6-11 |
| 6.4.3 System subroutine SWB | 6-11 |
| 6.5 AD51 and PC CPU Reset | 6-12 |
| 6.6 Notes on BASIC Programming | 6-12 |
| 7. COMMUNICATION WITH PROGRAMMABLE CONTROLLER CPU | 7-1 ~ 7-49 |
| 7.1 General-Purpose I/O Read/Write | 7-1 |
| 7.1.1 General-purpose I/O addresses | 7-2 |
| 7.1.2 Write to general-purpose input | 7-3 |
| 7.1.3 Read from general-purpose output | 7-5 |
| 7.2 Read/Write of Buffer Memory | 7-6 |
| 7.2.1 Read/write with BASIC program | 7-6 |
| 7.2.2 Read/write with sequence program | 7-7 |
| 7.3 Device Memory Read/Write | 7-9 |
| 7.3.1 System subroutines and device ranges | 7-9 |
| 7.3.2 BASIC program examples | 7-12 |
| 7.4 Extension File Register Read/Write | 7-22 |
| 7.4.1 System subroutines and functions | 7-22 |
| 7.4.2 BASIC program examples | 7-26 |
| 7.5 Special Function Module Buffer Memory Read/Write | 7-29 |
| 7.5.1 System subroutines and functions | 7-29 |
| 7.5.2 BASIC program examples | 7-33 |
| 7.6 Read/Write of Sequence Program and T/C Set Values | 7-34 |
| 7.6.1 System subroutines and functions | 7-34 |
| 7.6.2 Read and write procedures | 7-36 |
| 7.6.3 BASIC program example | 7-37 |

| | | |
|-------|--|------|
| 7.7 | Microcomputer Program Read/Write | 7-40 |
| 7.7.1 | System subroutines and functions | 7-40 |
| 7.7.2 | BASIC program examples | 7-41 |
| 7.8 | Comment Read/Write | 7-43 |
| 7.8.1 | System subroutines and functions | 7-43 |
| 7.8.2 | BASIC program examples | 7-44 |
| 7.9 | Interrupt from PC CPU to AD51 | 7-46 |
| 7.10 | Interrupt from AD51 to PC CPU | 7-47 |
| 7.11 | Remote RUN/STOP of PC CPU | 7-48 |

8. TROUBLESHOOTING 8-1 ~ 8-12

| | | |
|-------|----------------------------------|------|
| 8.1 | Screen Error Messages | 8-1 |
| 8.2 | Error Code List | 8-3 |
| 8.3 | Troubleshooting | 8-6 |
| 8.3.1 | Troubleshooting flow chart | 8-6 |
| 8.3.2 | Start-up troubleshooting | 8-7 |
| 8.3.3 | FDD troubleshooting | 8-8 |
| 8.3.4 | ROM troubleshooting | 8-9 |
| 8.3.5 | Multi task troubleshooting | 8-10 |
| 8.3.6 | CRT troubleshooting | 8-11 |
| 8.3.7 | Print-out troubleshooting | 8-12 |

9. MAINTENANCE 9-1 ~ 9-3

| | | |
|-----|----------------------------------|-----|
| 9.1 | Battery Life | 9-1 |
| 9.2 | Battery Changing Procedure | 9-2 |

APPENDICES APP-1 ~ APP-12

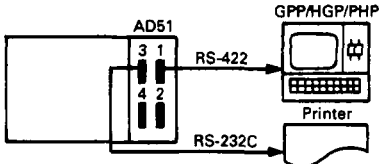
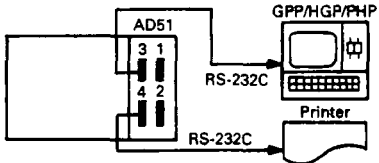
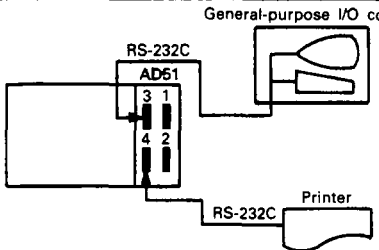
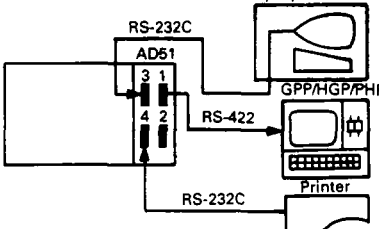
| | | |
|------------|---|--------|
| APPENDIX 1 | Differences between AD51-S3 and AD51 | APP-1 |
| APPENDIX 2 | Special Function Module Buffer Memory Address Tables | APP-2 |
| APPENDIX 3 | GPP/HGP Display Control Codes | APP-7 |
| APPENDIX 4 | GPP/HGP/PHP Key Codes and Character Codes | APP-8 |
| APPENDIX 5 | Storing the AD51E Memory Data into ROM Using the A6WU | APP-11 |
| APPENDIX 6 | External View | APP-12 |

1. INTRODUCTION

The AD51-S3 intelligent communication module (referred to as "AD51") has two RS-232C and two RS-422 interfaces and allows multitask processing of BASIC programs. User application programs running in the AD51 allow the following functions:

- (1) The A series peripheral devices can be shared.

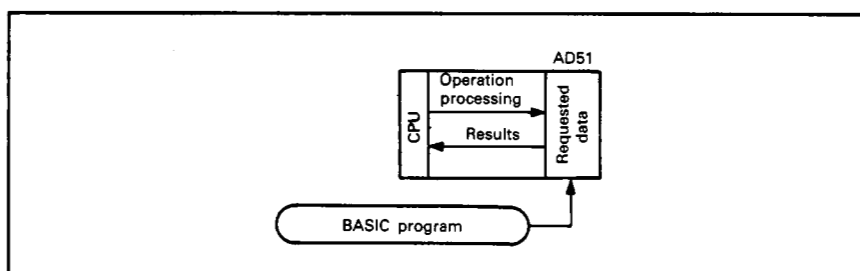
The A6GPP intelligent graphic programming panel (GPP), A6HGP LCD handy graphic programmer (HGP) or A6PHP plasma handy graphic programmer (PHP) started up by the SW-AD51P system disk (SW-AD51P) can be used the an I/O console of the AD51 and allows AD51 programs and data to be stored on disk (and ROM (GPP only)). The VT-220 terminal can also be used as an I/O console. The above indicated units may be connected as shown below:

| Connection Example | BASIC Program, Data | | |
|---|--|-----------------|------------------------------|
| | Input | Storage on disk | Storage on ROM |
|  | GPP/HGP/PHP | GPP/HGP/PHP | GPP (disallowed for HGP/PHP) |
|  | GPP/HGP/PHP | Disallowed | Disallowed |
|  | General-purpose I/O console | Disallowed | Disallowed |
|  | General-purpose I/O console GPP/HGP/PHP | GPP/HGP/PHP | GPP (disallowed for HGP/PHP) |

- (2) Can be used as a sub-CPU of the PC CPU.

The AD51 reads data (e.g. complicated numerical operation, function operation) from the PC and calculates and stores it as required so that the PC CPU is relieved from processing burden.

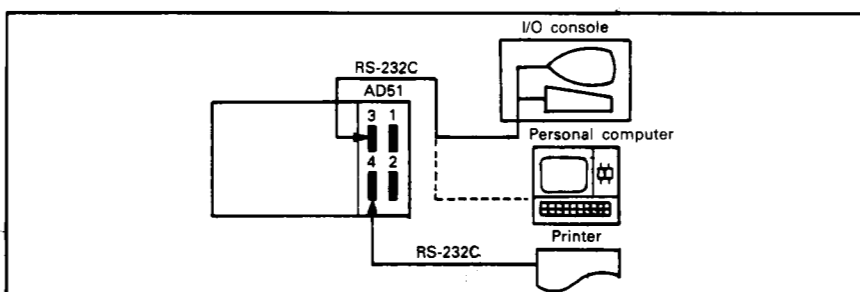
- 1) Storage of set value, positioning data, etc.
- 2) Collection, analysis and compensation of measurement data
- 3) Function operation of sine, log, square root, etc.
- 4) Logging and storage of production data
- 5) Logging and analysis of inspection data



- (3) Can be used as a monitoring module.

Connected with an I/O console (GPP/HGP/PHP or general-purpose I/O console), personal computer and printer, the AD51 allows the operating status to be monitored and control data to be printed out.

- 1) Monitor display Indicates the production status, operating conditions, fault definition, etc.
- 2) Keyboard entry Inputs the production schedule, production quantity, operating procedure and data setting.
- 3) Print out Prints out the production program, production results, daily report, fault definition, program data, inspection results and test results.
- 4) Clock function The AD51 includes an on-board, 24-hour, real-time clock with leap year compensation which allows data to be transferred by the BASIC program so that the time-of-the-day data may be processed easily.

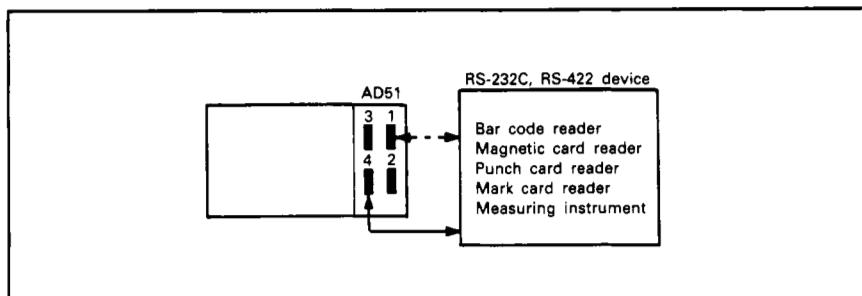


(4) Data input from bar code reader, etc.

The AD51 allows data to be entered from the bar code reader, magnetic card reader, etc. via RS-232C or RS-422.

The AD51 can be matched with the protocol of the connected device for communication by the BASIC program.

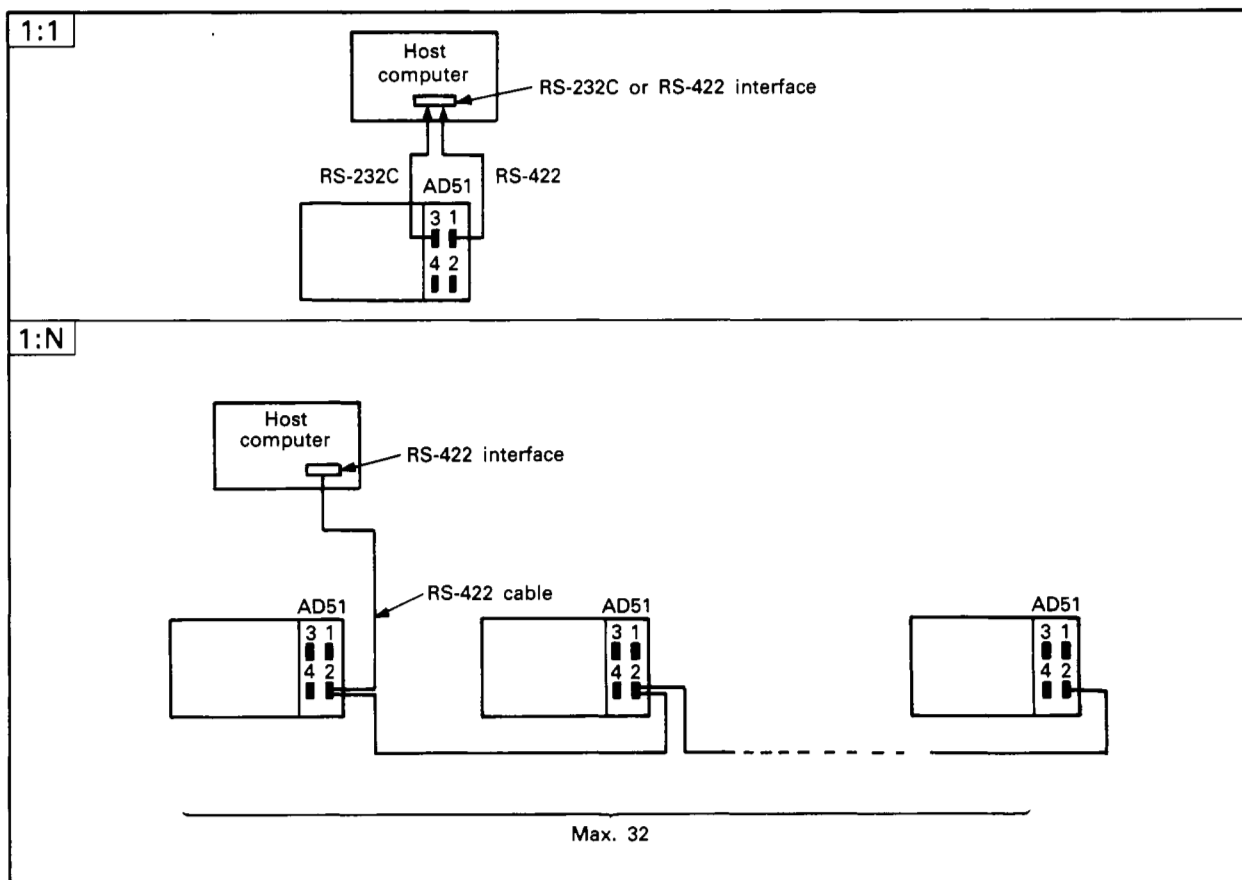
- 1) Entry of production lot number, product name, quantity, etc
- 2) Collection of measured values, test data.



(5) Connection of AD51 and external device

The AD51 has one RS-422 and two RS-232C interfaces for 1:1 link with the external device, and one RS-422 for 1:N multidrop link.

The application programs running in the AD51 allows data to be transferred via the four channels by the sequence program. The following examples show 1:1 and 1:N link configurations of the computer and AD51s:

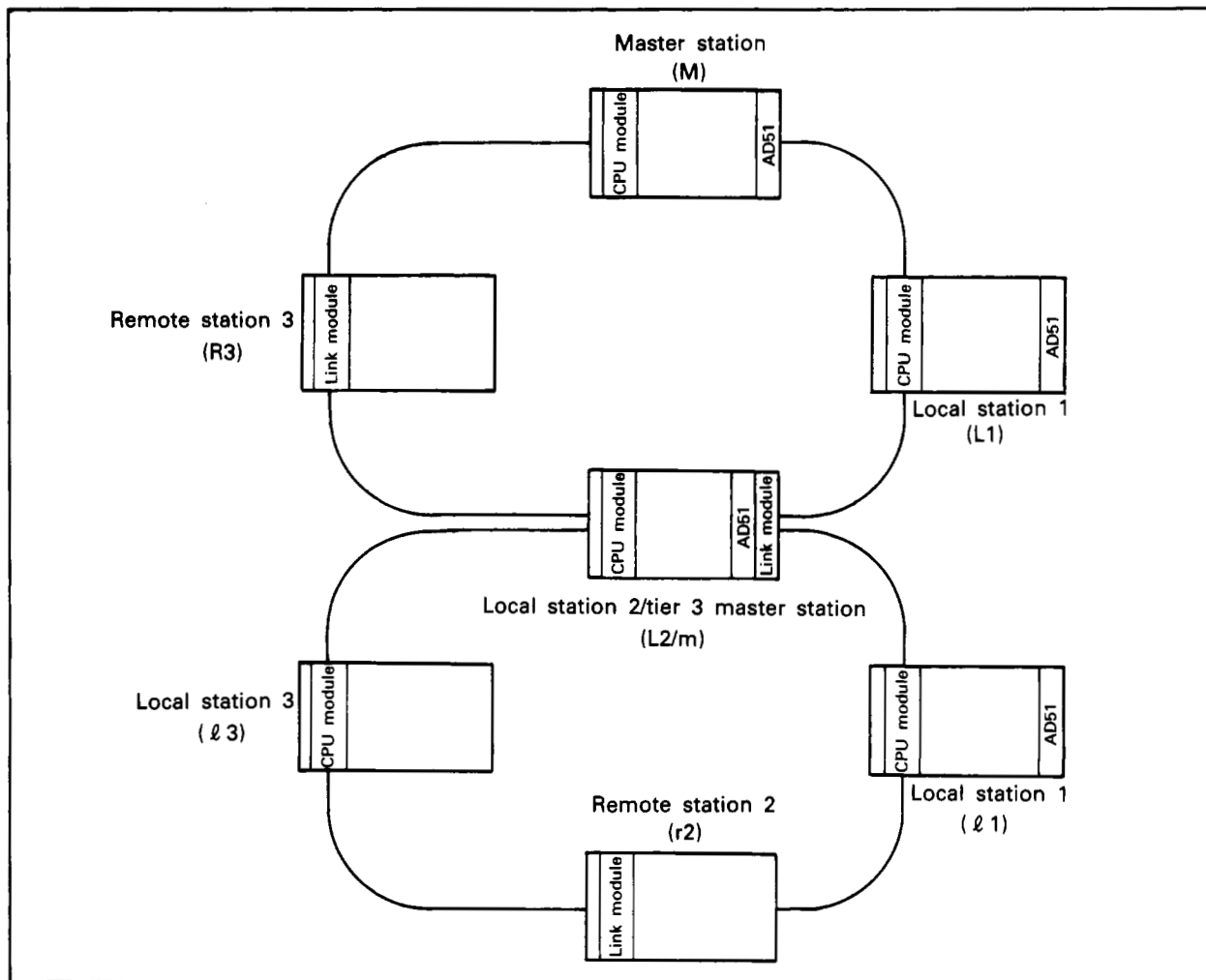


POINT

- (1) In the 1:1 link configuration, RS-422 is used for distances up to 500m and RS-232C for distances up to 15m.
- (2) In the 1:N link configuration, the RS-232C port may be connected with the host computer by using an RS-232C/RS-422 converter between the host computer and AD51.

(6) Communication with other stations in MELSECNET

The AD51 loaded on the PC CPU in MELSECNET allows communication with the other stations.



PC CPUs allowed for communication

(PC used with AD51)

- Station M (master station)
- Station L (local station)
- Station L/m (local station/tier 3 master station)
- Station (tier 3 local station)

(MELSECNET station allowed for communication)

- (1) Host station
- (2) All local stations in tier 2 (L1, L2/m)
- (3) Remote I/O station used with special function module in tier 2 (R3)
- (1) Host station
- (2) Master station in tier 2 (station M)
- (1) Host station
- (2) Master station in tier 2 (station M)
- (3) All local stations in tier 3 (L1, L3)
- (4) Remote I/O station used with special function module in tier 3 (r2)
- (1) Host station
- (2) Master station in tier 3 (L2/m)

1.1 Notes on Character Sets

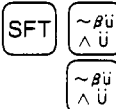
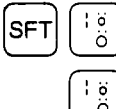
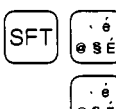
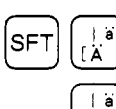

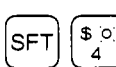

In this manual some of the characters used may differ from those which appear on the screen, depending on the character set chosen (i.e. Japanese, English, German, Swedish).

The keyboard operations follow the standard for the character set chosen, so, for example to input @ with English characters press

SFT , @ .

Key codes are given in Appendix 4, paragraph 3.

Key operations for the different character sets are shown below.

| Key | Character Set | | | |
|---|---------------|--------|---------|----------|
| | English | German | Swedish | Japanese |
| SFT  | ^ | ^ | Ü | ~ |
| | ~ | ß | ü | ^ |
| SFT  | \/ | Ö | Ö | |
| | | ö | ö | \/ |
| SFT  | @ | § | É | ` |
| | ` | ` | é | @ |
| SFT  | [| Ä | Ä | { |
| | { | ä | ä | [|
| SFT  |] | Ü | Å | |
| | } | ü | å |] |
| SFT  | £ | # | # | # |
| SFT  | \$ | \$ | ö | \$ |

In this manual, all examples use the Japanese character set.

2. SYSTEM CONFIGURATION

2.1 Overall Configuration

(1) Building block type PC

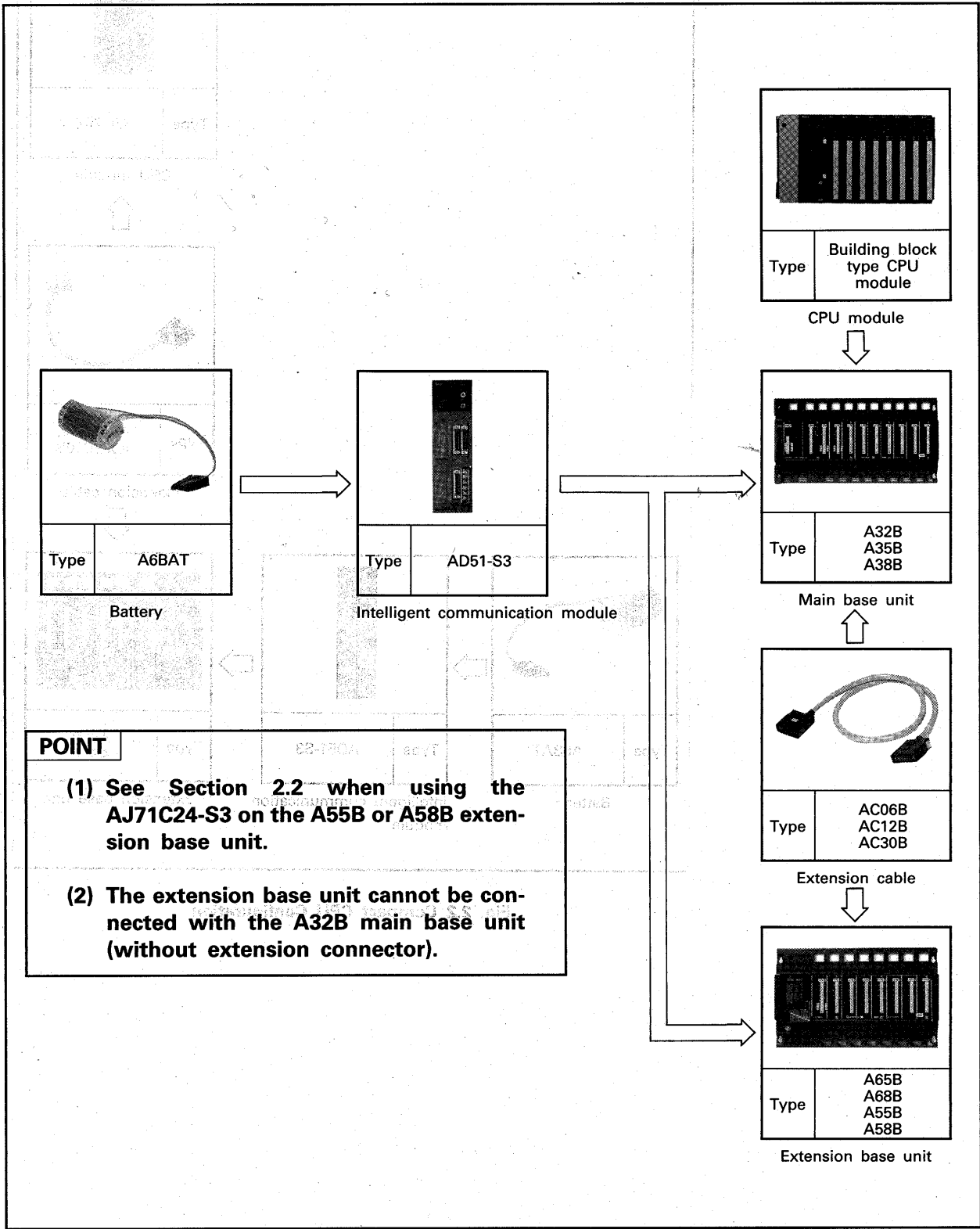


Fig. 2.1 Building Block CPU Configuration

(2) Compact type PC

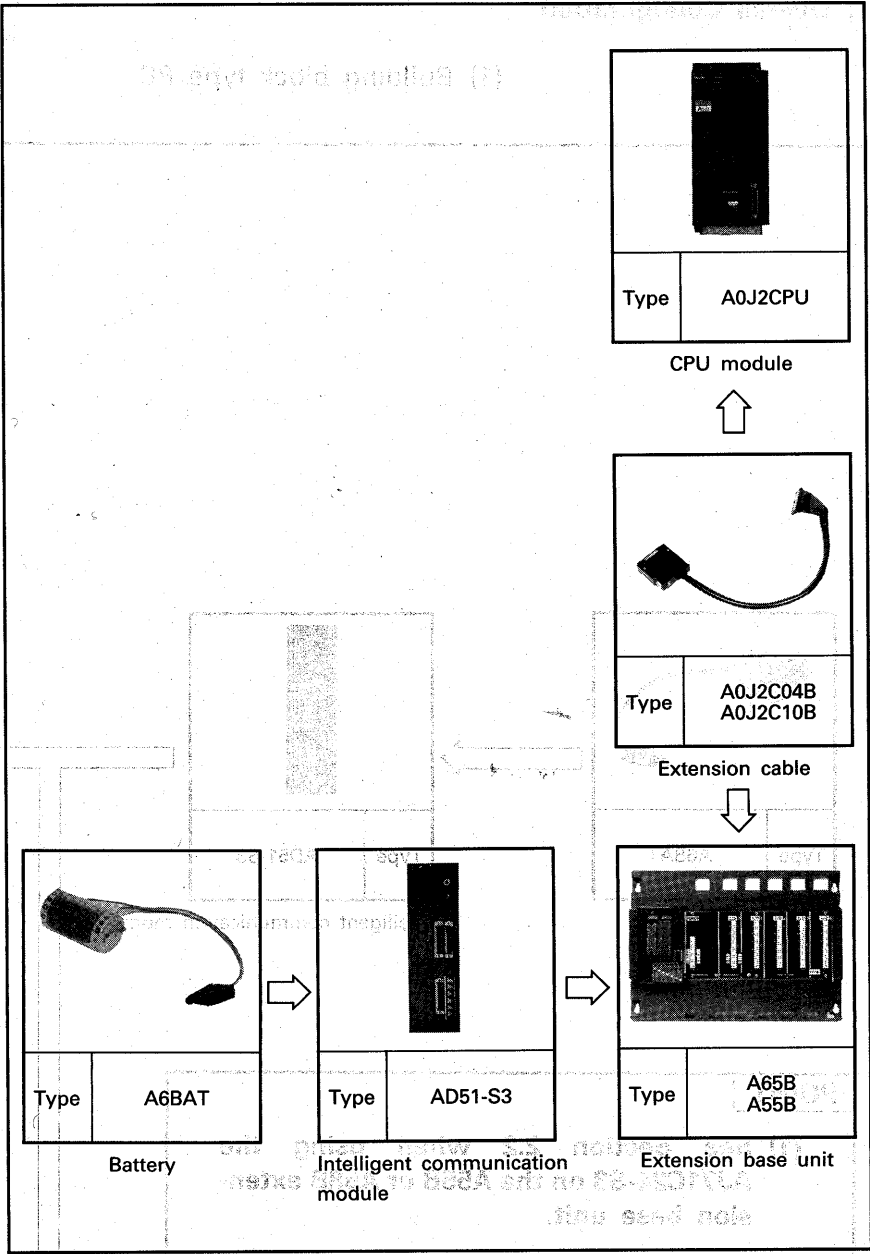


Fig. 2.2 Compact CPU Configuration

(3) Peripherals

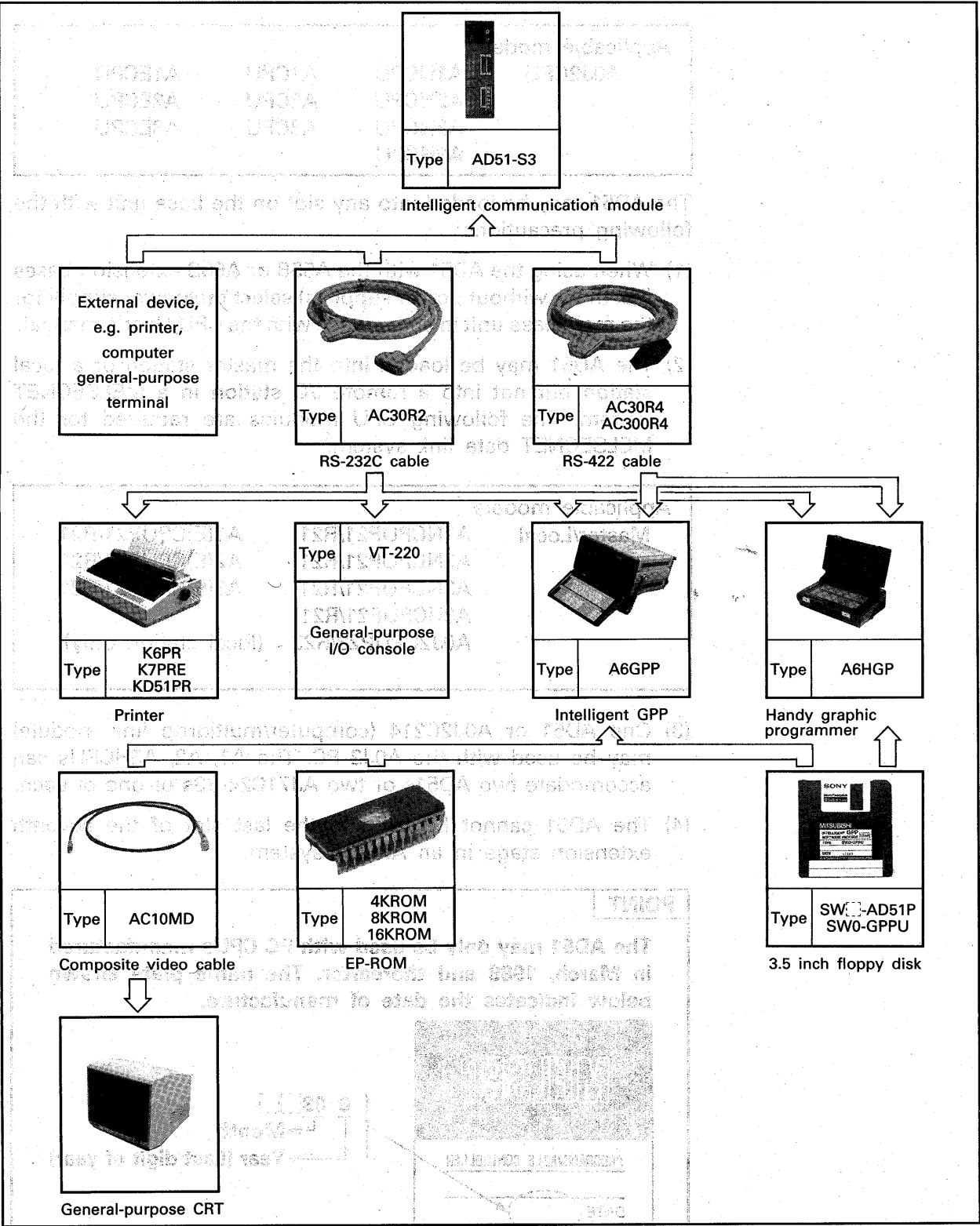


Fig. 2.3 Peripheral Device Configuration

POINT

The console select switch must be set to determine which programming terminal is to be used.

2. SYSTEM CONFIGURATION

MELSEC-A

2.2 Applicable A Series Systems

The AD51 can be used with the following CPU modules.

| Applicable models | | | |
|-------------------|-------|-------|-------|
| A0J2CPU | A1NCP | A1CPU | A1ECP |
| | A2NCP | A2CPU | A2ECP |
| | A3NCP | A3CPU | A3ECP |
| | A3HCP | | |

The AD51 may be loaded into any slot on the base unit with the following precautions:

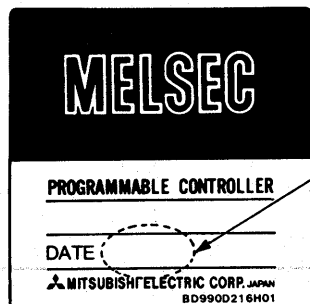
- (1) When using the AD51 with the A55B or A58B extension bases (i.e. those without power supplies) select the power supply for the main base unit in accordance with the CPU User's manual.
- (2) The AD51 may be loaded into the master station or a local station but not into a remote I/O station in a MELSECNET system. The following CPU modules are required for the MELSECNET data link system.

| Applicable models | | |
|----------------------|----------------|----------------------|
| Master/Local station | A1NCPUP21/R21 | A1(E)CPUP21/R21 |
| | A2NCPUP21/R21 | A2(E)CPUP21/R21 |
| | A3NCPUP21/R21 | A3(E)CPUP21/R21 |
| | A3HCPUP21/R21 | |
| | A0J2CPUP23/R23 | (local station only) |
| | | |

- (3) One AD51 or A0J2C214 (computer/multidrop link module) may be used with the A0J2 PC. The A1, A2, A3HCPUs can accommodate two AD51s or two AJ71C24-S3s or one of each.
- (4) The AD51 cannot be used in the last slot of the seventh extension stage in an A3CPU system.

POINT

The AD51 may only be used with PC CPUs manufactured in March, 1986 and thereafter. The name plate shown below indicates the date of manufacture.



6 03
→ Month
→ Year (Last digit of year)

2.3 Peripheral Equipment

The following table lists the peripheral equipment suitable for use with the AD51.

| Description | Type | Remarks | | | | | | | | | | | |
|---|--|---|---|-------|---|---------------|----------------------|---------------|----------------------|----------|---------------------------------|--------|--|
| Intelligent communication module | AD51-S3 | Main unit RAM support battery supplied | | | | | | | | | | | |
| EP-ROM | 8KROM | 16K bytes, for channels 1 and 2 | | | | | | | | | | | |
| | 16KROM | 32K bytes, for channels 1 and 2 24K bytes available to AD51 For details, refer to Section 3.4.2 | | | | | | | | | | | |
| Battery | A6BAT | For ICRAM support and real time clock | | | | | | | | | | | |
| Intelligent GPP | A6GPPE-SET | ○ Consists of the following models: | | | | | | | | | | | |
| | | Type | Remarks | A6GPP | • Programming unit with CRT. • Equipped with ROM writer, FDD, and printer interface functions. | SW- GPPAEE/EG | A series system disk | SW- GPPKEE/EG | K series system disk | SW- GPPU | User disk (3.5 inch, formatted) | AC30R4 | Cable for connection of AD51 and A6HGP, ACPU, AJ71C24-S3 |
| | | Type | Remarks | | | | | | | | | | |
| | | A6GPP | • Programming unit with CRT. • Equipped with ROM writer, FDD, and printer interface functions. | | | | | | | | | | |
| | | SW- GPPAEE/EG | A series system disk | | | | | | | | | | |
| | | SW- GPPKEE/EG | K series system disk | | | | | | | | | | |
| | | SW- GPPU | User disk (3.5 inch, formatted) | | | | | | | | | | |
| AC30R4 | Cable for connection of AD51 and A6HGP, ACPU, AJ71C24-S3 | | | | | | | | | | | | |
| Note: -EE; English version, EG; Germany version | | | | | | | | | | | | | |
| Handy graphic programmer | A6HGPE-SET | ○ Consists of the following: | | | | | | | | | | | |
| | | Type | Remarks | A6HGP | • Programming unit with LCD. • Equipped with FDD and printer interface functions. | SW- HGPAEE/EG | A series system disk | SW- HGPKEE/EG | K series system disk | SW- GPPU | User disk (3.5 inch, formatted) | AC30R4 | Cable for connection of AD51 and A6HGP, ACPU, AJ71C24-S3 |
| | | Type | Remarks | | | | | | | | | | |
| | | A6HGP | • Programming unit with LCD. • Equipped with FDD and printer interface functions. | | | | | | | | | | |
| | | SW- HGPAEE/EG | A series system disk | | | | | | | | | | |
| | | SW- HGPKEE/EG | K series system disk | | | | | | | | | | |
| | | SW- GPPU | User disk (3.5 inch, formatted) | | | | | | | | | | |
| AC30R4 | Cable for connection of AD51 and A6HGP, ACPU, AJ71C24-S3 | | | | | | | | | | | | |
| Note: -EE; English version, EG; Germany version | | | | | | | | | | | | | |
| System disk | SW-AD51P | System software package for the A6GPP or A6HGP (back-up copy provided) | | | | | | | | | | | |
| User disk | SW0-GPPU | User disk (already formatted) for storing programs. | | | | | | | | | | | |
| Composite video cable | AC10MD | Optional cable for GPP external monitor 1m(3.28ft) length | | | | | | | | | | | |
| General-purpose I/O console | VT-220 | Display control codes equivalent to VT-220. | | | | | | | | | | | |
| Printer | K6PRE K7PRE | For program hard copy and data print out | | | | | | | | | | | |
| | KD51PR | For printing data | | | | | | | | | | | |
| RS-422 cable | AC300R4 | Cable between AD51 and A6GPP 3m(9.84ft) length | | | | | | | | | | | |
| RS-232C cable | AC30R2 | Connection cable between AD51 and printer and for VT-220. 3m(9.84ft) length | | | | | | | | | | | |

Table 2.1 System Equipment List (Continue)

| Description | Type | Remarks |
|---------------------|----------|---|
| Ink ribbon | K6PR-R | Ink ribbon for K6PRE |
| | K7PR-R | Ink ribbon for K7PRE |
| | KD51PR-R | Ink ribbon for KD51PR |
| Printer paper | K6PR-Y | Printer paper for K6PR |
| | KD51PR-Y | Printer paper for KD51PR |
| Interface connector | 232-CON | Connector for RS-422 and RS-232C interfaces |

Table 2.1. System Equipment List

3. SPECIFICATIONS

3.1 Performance Specifications

| Item | | Specification |
|-----------------------------------|---------|--|
| Processor | | HD64180 |
| Program language | | GPC-BASIC |
| Number of tasks | | Maximum 8 |
| Task start conditions | | Power on |
| | | Interrupt from PC CPU |
| | | Real time interrupt (Set in the range 0.01 to 9.99 seconds in units of 0.01 second.) |
| Internal memory | | Maximum: 114K bytes=64K bytes+2K bytes+48K bytes Common work area Two 16KROMs loaded *For details, refer to Section 3.4. |
| General-purpose I/O | | General-purpose input : 13 points General-purpose output: 10 points *For details, refer to Section 3.6. |
| Buffer memory | | 3K words (6K bytes) *For details, refer to Section 3.7. |
| Memory protect address range | | 4F00 to 4FFF (system data area) 8000 to DFFF (channel 1 to 4) |
| Interfaces | RS-422 | Conforms to EIA. RS-422. Channel 1: D shell connector. Channel 2: Terminal block Transmission distance: ≤500m |
| | RS-232C | Conforms to EIA. RS-232C. Channel 3, 4: D shell connector. Transmission distance: ≤15m |
| Arithmetic and logic unit (ALU) | | Performs high-speed processing of BASIC's intrinsic functions (trigonometric, inverse trigonometric, logarithm, exponential, √, absolute value). |
| Clock element | | Year, month, day, hour, minute, second Read/write 24 hour mode, automatic leap year compensation |
| Power failure compensation | | Internal memory, lithium battery for back-up of real time clock Total back-up time: 130 days Battery life : 5 years |
| Console | | A6GPP, A6HGP, A6PHP VT-220 |
| Number of I/O points occupied | | 48 |
| Internal current consumption (5V) | | 1.3A |
| Size mm (inch) | | 250 (9.84) (H)×76 (2.99) (W)×120 (4.72) (D) |
| Weight kg (lb) | | 1.1 (2.42) |

Table 3.1 AD51 Performance Specifications

3.2 Instruction Set

The AD51 is programmed in GPC-BASIC.
In addition to the BASIC commands a series of subroutines are available which can be called from the BASIC program.

3.2.1 GPC-BASIC commands

The following table lists the GPC-BASIC commands. For full information refer to the "GPC-BASIC Handbooks". The graphics commands described in the "GPC-BASIC" Handbooks are not available on the AD51.

| Command | | Function |
|--------------------------------------|------------|---|
| Key command <div>KEY</div> | AUTO | Automatic generation of line number |
| | BYE | Returns to BASIC program address data screen. |
| | CONT | Resumes program run after BREAK |
| | COMPILE | Compiles multitask executable program |
| | DELETE | Deletes program from specified line number to specified line number |
| | EDIT | Corrects statement in one line |
| | EXECUTE | Run of program after "RUN" or "COMPILE" |
| | LIST | Displays program on screen |
| | | |
| | | |
| | LLIST | Prints out program |
| | NEW | Erases program |
| | RENUM | Renumbers line numbers |
| | | |
| | RUN | Executes program |
| | ZDV | Displays I/O console |
| | — | Deletes line |
| Program command <div>PRG</div> | BREAK | Resumes program run after stop and "CONT" |
| | CALL | Calls machine language program |
| | CLS | Clears CRT screen |
| | | |
| | CLOSE | Closes specified RS-232C/RS-422 channel. |
| | | |
| | | |
| | | |
| | END | Declares end of program run |
| | FOR...NEXT | Repeats program run from "FOR" to "NEXT" |
| | GOTO | Moves to specified line number |

Table 3.2 BASIC Command List (Continue)

| Command | | Function |
|------------------------|-----------------------------|---|
| | GOSUB RETURN | Moves to specified subroutine Returns from subroutine |
| | IF | Judges result of expression |
| INPUT | A | Input from keyboard |
| | B | |
| INKEY | | Assigns input from keyboard to variable |
| LET | | Assigns value of expression to variable |
| LOCATE | A | Moves cursor position |
| | B | |
| | C | |
| | D | |
| LPRINT | A | Prints out data |
| | B | |
| | C | |
| ONGOSUB | | Moves to subroutine in line number specified by value of expression |
| ONGOTO | | Moves to line number specified by value of expression |
| Program command PRG | A | Opens specified RS-232C/RS-422 channel. |
| | B | |
| | C | |
| | D | |
| PEEK | | Reads 1-byte data from specified memory address |
| POKE | | Writes 1-byte data to specified memory address |
| PRINT | A | Displays data on screen |
| | B | |
| REM | | Used to write comment. |
| SIZE | A | Displays text program capacity |
| | B | |
| STOP | | Stops program run |
| ZCOFF | A | Underline Type A when the console is GPP/HGP. Type B when the console is VT-220. |
| | B | |
| ZCON | A | Resets underline (used after "ZCOFF"). Type A when the console is GPP/HGP. Type B when the console is VT-220. |
| | B | |
| ZCRV | | Reverses character color on CRT screen |
| ZDATE | | Reads year, month, day, hour, and minute |
| ZIDV | A | Changes input console |
| | B | |
| | C | |
| | D | |
| ZMOV | | Transfers data from memory to memory |
| ZNOR | | Returns the character reversed after "ZCRV" to its original color |

Table 3.2 BASIC Command List (Continue)

| Command | | | Function |
|-----------------------------|--------------------------------------|---|--|
| Program command [PRG] | ZODV | A | Changes output console |
| | | B | |
| | | C | |
| | ZRD1 | | Reads 1-byte data from specified channel |
| | ZRD2 | | Reads 2-byte data from specified channel |
| | ZTIME | | Stops execution for specified interval of time |
| | ZWR1 | | Writes 1-byte data to specified channel |
| | ZWR2 | | Writes 2-byte data to specified channel |
| Intrinsic function [INT] | ABF | | Absolute value of real number in mathematical expression |
| | ABS | | Absolute value of integer in mathematical expression |
| | ACOS | | Arccosine (\cos^{-1}) of mathematical expression |
| | ASIN | | Arcsine (\sin^{-1}) of mathematical expression |
| | ATAN | | Arctangent (\tan^{-1}) of mathematical expression |
| | COS | | Cosine (cos) of mathematical expression |
| | EXP | | Value of Exponential to base "e" ($e=2.718281$) |
| | LN | | Value of natural logarithm (loge) |
| | LOG | | Value of common logarithm (\log_{10}) |
| | NOT | | Generates "1" when value of mathematical expression is "0" and generates "0" when the value is not "0". |
| | RND | | Assigns random number to variable |
| | SIN | | Sine (sin) of mathematical expression |
| | SQRT | | Value of square root of mathematical expression |
| | TAN | | Tangent (tan) of mathematical expression |
| | Arith- metic operator [ALU] | + | Addition |
| | | - | Subtraction |
| | | × | Multiplication |
| | | / | Division |
| | | ^ | Exponent |
| | | - | Sign reversion |
| | | % | Remainder calculation |

Table 3.2 BASIC Command List (Continue)

| Command | | Function |
|---|----|------------------------------|
| Com- parison operator <div>COM</div> | = | Is equal to |
| | # | Is not equal to |
| | < | Is less than |
| | > | Is greater than |
| | <= | Is not greater than |
| | >= | Is not less than |
| Logical operator <div>LOG</div> | # | Negation (NOT) |
| | & | Logical product (AND) |
| | | Logical sum (OR) |
| | ⊕ | Exclusive logical sum (EXOR) |

Table 3.2 BASIC Command List

Where commands have several options (indicates as

A

,

B

,

C

 etc.) only those shaded () may be used on the AD51.

3.2.2 System subroutines

System subroutines are machine code programs used for special AD51 functions (for example PC CPU transactions etc.). They are already written in channel 0 of the AD51 at specified address locations.

System subroutine operation is initiated by using the "CALL" command in the BASIC program.

System subroutines on the AD51 are shown in Table 3.3.

[Initializing the system subroutine]

- 1) The system subroutine is called from the GPC-BASIC program using the "CALL" command.
- 2) The format of the CALL statement is as follows.

A=CALL (variable 1, variable 2, [variable 3, variable 4])

Variable 1: Always 0. All system subroutines are located in channel 0.

Variable 2: Head address of system subroutine in channel 0 (see table 3.3)

Variable 3: Variable for system subroutine stored in (D)(E) registers.

Variable 4: Variable for system subroutine stored in (B)(C) registers.

- 3) For information on variable 3 and variable 4, refer to the GPC-BASIC Handbooks.
- 4) Before executing the CALL command, transfer variables to the work area.

| | System Subroutine | Function | Channel | Address |
|----|-------------------|---|---------|-------------------|
| 1 | SAI | ASCII (hexadecimal) → BIN | 0 | 8060 _H |
| 2 | SIA | BIN → ASCII (hexadecimal) | 0 | 8063 _H |
| 3 | SAN | ASCII (decimal) → BIN | 0 | 8072 _H |
| 4 | SNA | BIN → ASCII (decimal) | 0 | 8075 _H |
| 5 | SAF | ASCII → real number | 0 | 8066 _H |
| 6 | SFA | Real number → ASCII | 0 | 8069 _H |
| 7 | SBF | Integer → real number | 0 | 806C _H |
| 8 | SFB | Real number → integer | 0 | 806F _H |
| 9 | SFLTD | 32-bit integer → 32-bit floating point number | 0 | 80DE _H |
| 10 | SFIXD | 32-bit floating point number → 32-bit integer | 0 | 80E1 _H |
| 11 | SBD4 | BIN → 4-digit BCD | 0 | 8042 _H |
| 12 | SDB4 | 4-digit BCD → BIN | 0 | 8045 _H |
| 13 | SBD6 | BIN → 6-digit BCD | 0 | 8048 _H |
| 14 | SDB6 | 6-digit BCD → BIN | 0 | 804B _H |

Table 3.3 System Subroutine List (Continue)

| | System Subroutine | Function | Channel | Address |
|----|-------------------|---|---------|-------------------|
| 15 | SBA | BIN addition (24 bits) | 0 | 804E _H |
| 16 | SBS | BIN subtraction (24 bits) | 0 | 8051 _H |
| 17 | SBM | BIN multiplication (24 bits) | 0 | 8054 _H |
| 18 | SBW | BIN division (24 bits) | 0 | 8057 _H |
| 19 | SCA | Write to clock element (year, month, day, hour, minute, second) | 0 | 803C _H |
| 20 | SCB | Read from clock element (year, month, day, hour, minute, second) | 0 | 803F _H |
| 21 | SPC | Discrimination of programmable controller CPU | 0 | 8078 _H |
| 22 | SKC | Programmable controller CPU run/stop check | 0 | 8030 _H |
| 23 | SKR | Remote run of programmable controller CPU | 0 | 8033 _H |
| 24 | SKP | Remote stop of programmable controller CPU | 0 | 8036 _H |
| 25 | SRB | Receives specified byte length of data sent to specified channel | 0 | 8009 _H |
| 26 | SWB | Sends specified byte length of data from specified channel | 0 | 800C _H |
| 27 | SRC | Reads the number of bytes of data received by specified channel | 0 | 800F _H |
| 28 | SRF | Reads the number of vacant bytes in receive buffer of specified channel | 0 | 8012 _H |
| 29 | SHX | Controls send/receive data of specified channel by Xon/Xoff codes | 0 | 8015 _H |
| 30 | SHD | Controls send/receive data of specified channel by DR terminal | 0 | 8018 _H |
| 31 | SAE | Converts all send/receive data of specified channel to EBCDIC code | 0 | 801E _H |
| 32 | SEA | No code conversion of send/receive data of specified channel | 0 | 8021 _H |
| 33 | STC | Reads the number of remaining bytes in send buffer of specified channel | 0 | 801B _H |
| 34 | SRP | Reads status of specified channel | 0 | 8027 _H |
| 35 | SR2 | Reads data from buffer memory | 0 | 8000 _H |
| 36 | SW2 | Writes data to buffer memory | 0 | 8003 _H |
| 37 | SADR | Reads data from data memory of programmable controller CPU | 0 | 807B _H |
| 38 | SADW | Writes data to data memory of programmable controller CPU | 0 | 807E _H |
| 39 | SADT | Randomly writes data to data memory of programmable controller CPU | 0 | 8081 _H |
| 40 | SADM0 | Enters data randomly read from data memory of programmable controller CPU | 0 | 8084 _H |

Table 3.3 System Subroutine List (Continue)

| | System Subroutine | Function | Channel | Address |
|----|-------------------|--|---------|-------------------|
| 41 | SADM1 | Randomly reads data from data memory of programmable controller CPU | 0 | 8087 _H |
| 42 | SAAR | Reads sequence program | 0 | 808A _H |
| 43 | SAAW | Writes sequence program | 0 | 808D _H |
| 44 | SAPR | Reads programmable controller CPU parameters | 0 | 8090 _H |
| 45 | SAPW | Writes programmable controller CPU parameters | 0 | 8093 _H |
| 46 | SAPS | Analysis request of programmable controller CPU parameters | 0 | 8096 _H |
| 47 | SIT | Interruption to programmable controller CPU | 0 | 802A _H |
| 48 | SIR | Reads error code | 0 | 8024 _H |
| 49 | SC2 | Sets SR2/SW2 retry time | 0 | 8006 _H |
| 50 | *1 SAER | Reads data from extension file registers of PC CPU | 0 | 80BD _H |
| 51 | *1 SAEW | Writes data to extension file registers of PC CPU | 0 | 80C0 _H |
| 52 | *1 SAET | Randomly writes data to extension file registers of PC CPU | 0 | 80C3 _H |
| 53 | *1 SAEM0 | Defines PC CPU extension file registers to be monitored | 0 | 80C6 _H |
| 54 | *1 SAEM1 | Monitors PC CPU extension file registers specified in monitor data entry | 0 | 80C9 _H |
| 55 | *2 SAMR | Reads microcomputer program from PC CPU | 0 | 80CC _H |
| 56 | *2 SAMW | Writes microcomputer program to PC CPU | 0 | 80CF _H |
| 57 | SACR | Reads comments from PC CPU | 0 | 80D2 _H |
| 58 | SACW | Writes comments to PC CPU | 0 | 80D5 _H |
| 59 | SATR | Reads data from special function module buffer memory | 0 | 80D8 _H |
| 60 | SATW | Writes data to special function module buffer memory | 0 | 80DB _H |

Table 3.3 System Subroutine List

Where system subroutines have several options (indicated as A, B) only those shaded () may be used on the AD51.

REMARKS

- (1) System subroutines indicated as —
B are covered in the GPC-BASIC Handbooks.
- (2) System subroutines marked *1 may only have access to the A3ECPU, A3CPU, A3NCPU and A3HCPU.
System subroutines marked *2 may only have access to the A1ECPU, A1CPU, A1NCPU, A2ECPU, A2CPU, A2NCPU, A3ECPU, A3CPU, A3NCPU and A3HCPU.

3.3 Software Configuration

The following shows a block diagram of the AD51 software configuration indicating how the various areas interact.

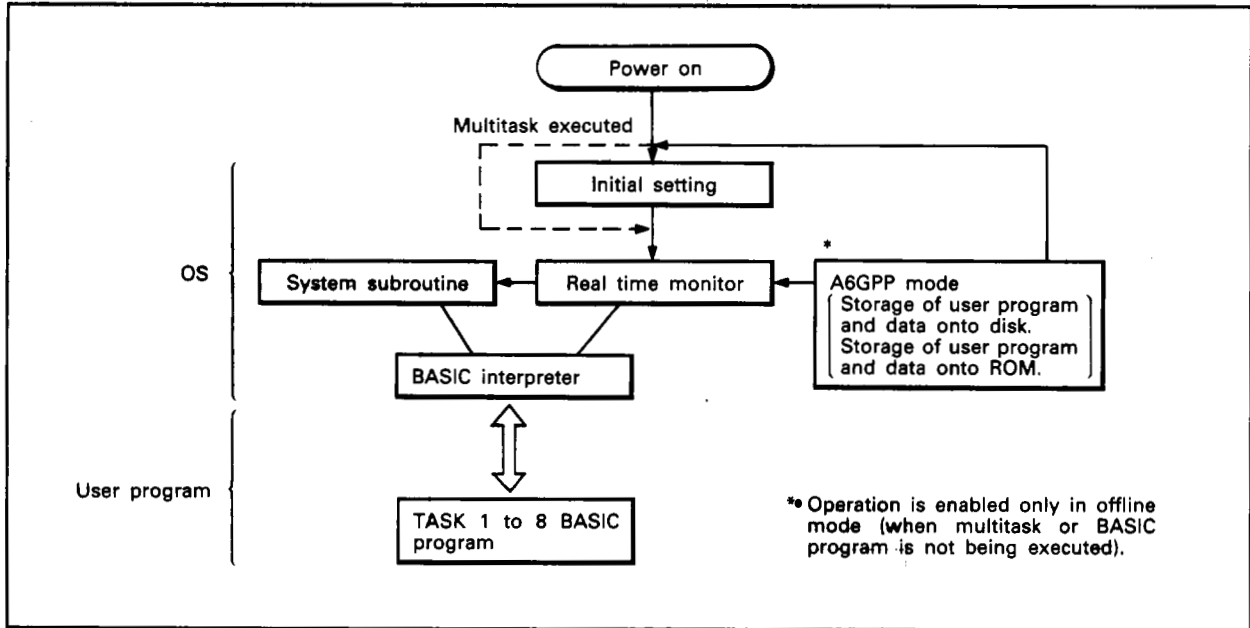


Fig. 3.1 Software Configuration

- (1) As shown in Fig. 3.1, a maximum of 8 user programs may be processed in parallel under the control of the real time monitor.
- (2) "Power on", "interrupt from ACPU", and "real time interrupt" are available as starting conditions for the user program.
- (3) Each task can only be written in BASIC.

3.4 Memory Map

3.4.1 Memory configuration

The AD51 is a Z-80 based system. To expand the memory size from 64K bytes the second 32K bytes are duplicated in additional channels as shown in Fig. 3.2 below.

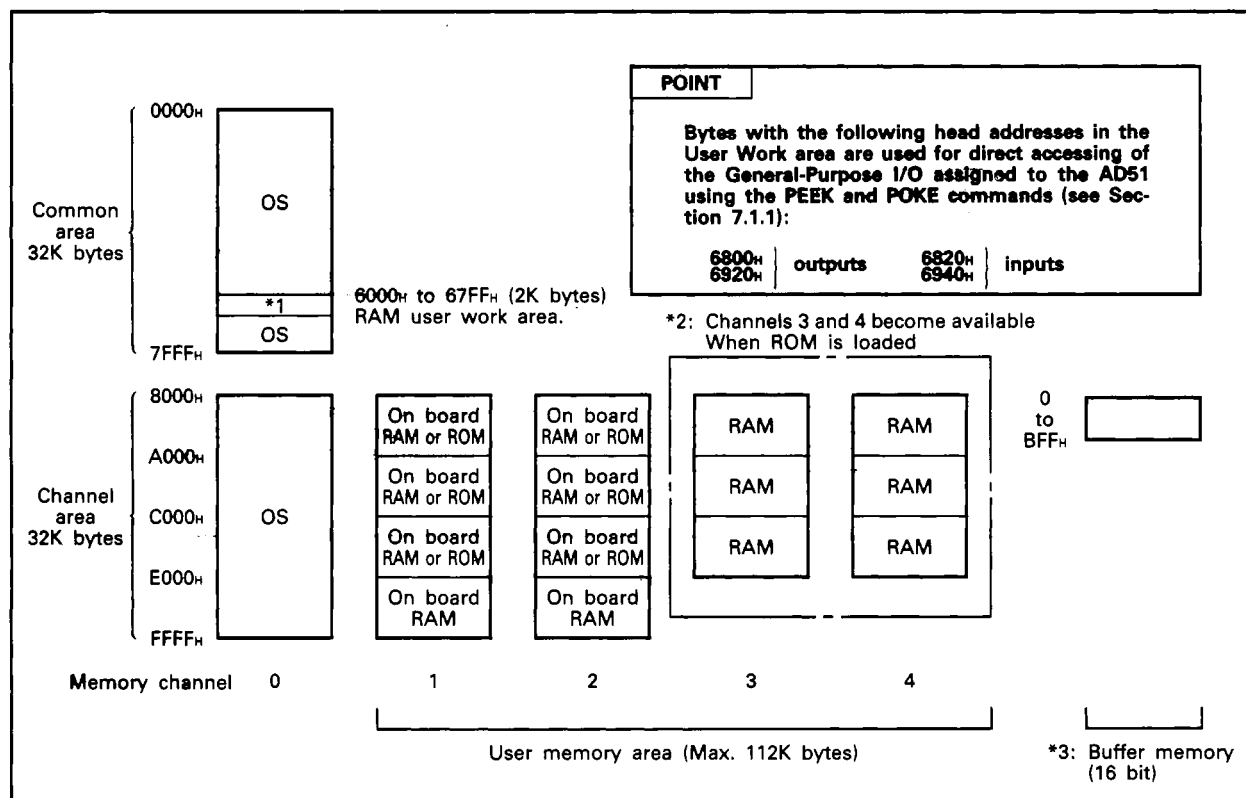


Fig. 3.2 Memory Map

POINT

- *(1) The RAM user work area, (addresses 6000H to 67FFH) is in the common area and can be accessed by the user programs in any of channels 1 to 4.
- *(2) The memory area may be expanded by adding ROM to channels 1 and/or 2.
- *(3) The buffer address range is 000H to BFFH which represents 3K words (6K bytes) of buffer memory. Each buffer memory address represents 1 word. (i.e. 16 bits)

3.4.2 Memory map with ROM loaded

The user memory area can be expanded by loading ROM into channels 1 and/or 2. The memory map will vary as shown below depending on the location and size of the ROMs.
8K and 16KROMs may be used.

| Channel 1 Channel 2 | | Without ROM | 8KROM | 16KROM |
|------------------------|---|---|---|---|
| Without ROM | 8000H A000H C000H E000H FFFFH | <div>RAM : 66K bytes ROM : 0K byte Total : 66K bytes</div> <div><div>RAM</div><div>RAM</div><div></div><div></div></div> <div>Channel 1234</div> | <div>RAM : 66K bytes ROM : 16K bytes Total : 82K bytes</div> <div><div>ROM</div><div>RAM</div><div>RAM</div><div></div></div> <div>Channel 1234</div> | <div>RAM : 66K bytes ROM : 24K bytes Total : 90K bytes</div> <div><div>ROM</div><div>RAM</div><div>RAM</div><div></div></div> <div>Channel 1234</div> |
| | | <div>RAM : 66K bytes ROM : 16K bytes Total : 82K bytes</div> <div><div>RAM</div><div>ROM</div><div></div><div>RAM</div></div> <div>Channel 1234</div> | <div>RAM : 66K bytes ROM : 32K bytes Total : 98K bytes</div> <div><div>ROM</div><div>ROM</div><div>RAM</div><div>RAM</div></div> <div>Channel 1234</div> | <div>RAM : 66K bytes ROM : 40K bytes Total : 106K bytes</div> <div><div>ROM</div><div>ROM</div><div>RAM</div><div>RAM</div></div> <div>Channel 1234</div> |
| | | <div>RAM : 66K bytes ROM : 24K bytes Total : 90K bytes</div> <div><div>RAM</div><div>ROM</div><div></div><div>RAM</div></div> <div>Channel 1234</div> | <div>RAM : 66K bytes ROM : 40K bytes Total : 106K bytes</div> <div><div>ROM</div><div>ROM</div><div>RAM</div><div>RAM</div></div> <div>Channel 1234</div> | <div>RAM : 66K bytes ROM : 48K bytes Total : 114K bytes</div> <div><div>ROM</div><div>ROM</div><div>RAM</div><div>RAM</div></div> <div>Channel 1234</div> |
| | | <div>RAM : 66K bytes ROM : 24K bytes Total : 90K bytes</div> <div><div>RAM</div><div>ROM</div><div></div><div>RAM</div></div> <div>Channel 1234</div> | <div>RAM : 66K bytes ROM : 40K bytes Total : 106K bytes</div> <div><div>ROM</div><div>ROM</div><div>RAM</div><div>RAM</div></div> <div>Channel 1234</div> | <div>RAM : 66K bytes ROM : 48K bytes Total : 114K bytes</div> <div><div>ROM</div><div>ROM</div><div>RAM</div><div>RAM</div></div> <div>Channel 1234</div> |

Table 3.4 Memory Map When ROM is Used.

POINT

- (1) The installation of a ROM shifts the corresponding RAM address range to a different channel. (e.g. Installing 8K of ROM at channel 1 addresses 8000_H to BFFF_H moves the RAM area to channel 3 addresses 8000_H to BFFF_H.)
- (2) Two shorting pins are used to specify RAM or ROM in channels 1 and 2. The RAM area is moved as follows depending on the pin setting in each channel.

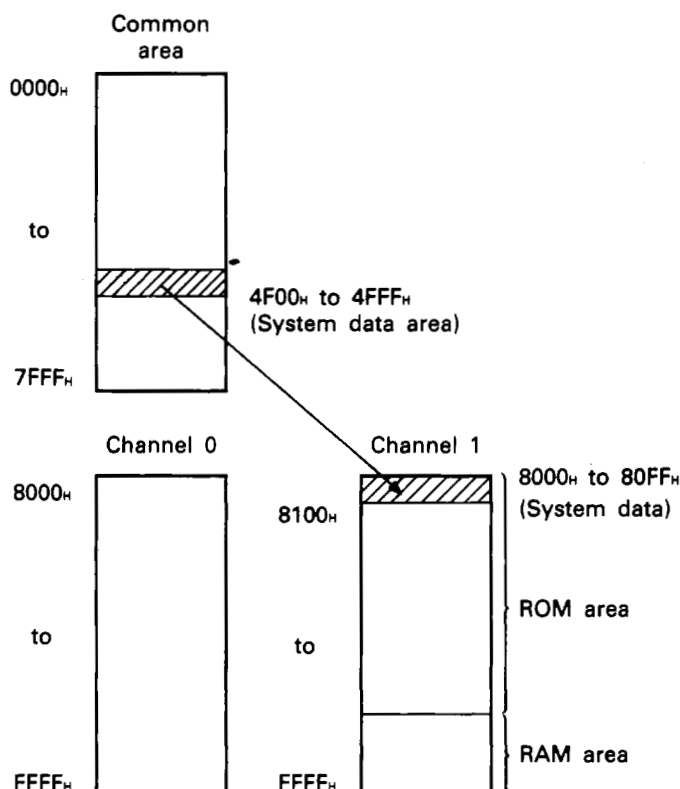
| RAM/ ROM setting pin | ROM loading | |
|-------------------------|-----------------------------|--|
| | With ROM | Without ROM |
| RAM position | RAM and ROM areas coincide. | Correct |
| ROM position | Correct | The RAM area changes channels as though ROM was installed. |

- (3) Only 24K bytes are valid (addresses 8000_H to DFFF_H) when 16KROM is used.
The 8K bytes from E000_H to FFFF_H are used in the RAM area and the program in this ROM address range cannot be executed.
- (4) For RAM area memory protect, refer to Section 4.3.1.

3.4.3 Storing system data on ROM

System data (e.g. multitask set data) required for program execution of each task may be stored to ROM.

The system data located in the common area is stored to the first 256-byte area of the ROM installed on channel 1.



- (1) After the system data is stored on ROM, addresses 8100_H to $FFFF_H$ are used as a user memory area and 8000_H to $80FF_H$ cannot be used as a user area (program area, work area).
- (2) After the system data is stored on ROM, the RAM area of channel 4 can be used as a program area but that of channel 3 may only be used as a work area.
- (3) With the system data transfer switch ON, multitask is executed after the system data is transferred from 8000_H to $80FF_H$ of channel 1 to the system data area ($4F00_H$ to $4FFF_H$) at power on.

3. SPECIFICATIONS



3.5 Interface Specifications

3.5.1 RS-422 connector specifications (CH1)

| Item | Specifications |
|---------------------|---|
| Connected Unit | A6GPP, printer with RS-422, personal computer, etc. |
| Transmission system | Conforms to EIA. RS-422. |
| Synchronous system | Asynchronous system |
| USART mode setting | <div>Baud rate setting (300, 600, 1200, 2400, 4800, 9600 BPS selectable)</div> <div>Parity bit setting — Parity absent — Even parity</div> <div>Parity bit setting — Parity present — Odd parity</div> <div>Stop bit setting — Stop bit 1</div> <div>Stop bit setting — Stop bit 2</div> <div>Character data bit setting — Data 7 bits</div> <div>Character data bit setting — Data 8 bits</div> <div>Communication control setting — XON/XOFF control</div> <div>Communication control setting — Control with DTR terminal</div> |

| Connector pin outs. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|--|---|-----------------------|------------------|---|------------------|---------|-----------|-----|-------|-----------|--|-----|-------|--------------|-----|-------|-----------|--|-----|-------|---------------------|-----|-------|-----------|--|-----|-------|----------------|-----|-------|-----------|--|-----|-------|------------|--|-----------------------|--|-----------------------|---------------|-----|-------------|--|---|-----|---|--------------|----|---|--|--|
| <div>1 ○</div> <div>2 ○</div> <div>3 ○</div> <div>4 ○</div> <div>5 ○</div> <div>6 ○</div> <div>7 ○</div> <div>8 ○</div> <div>9 ○</div> <div>10 ○</div> <div>11 ○</div> <div>12 ○</div> <div>13 ○</div> <div>14 ○</div> <div>15 ○</div> <div>16 ○</div> <div>17 ○</div> <div>18 ○</div> <div>19 ○</div> <div>20 ○</div> <div>21 ○</div> <div>22 ○</div> <div>23 ○</div> <div>24 ○</div> <div>25 ○</div> | <table><tr><th>Signal</th><th>Block Diagram</th><th>Pin</th><th>Signal Direction</th><th>Remarks</th></tr><tr><td rowspan="2">Send data</td><td>SDA</td><td>(+) ③</td><td rowspan="2">→ outside</td><td rowspan="2"></td></tr><tr><td>SDB</td><td>(-) ⑫</td></tr><tr><td rowspan="2">Receive data</td><td>RDA</td><td>(+) ②</td><td rowspan="2">← outside</td><td rowspan="2"></td></tr><tr><td>RDB</td><td>(-) ⑮</td></tr><tr><td rowspan="2">Data terminal ready</td><td>CSA</td><td>(+) ⑤</td><td rowspan="2">→ outside</td><td rowspan="2"></td></tr><tr><td>CSB</td><td>(-) ⑱</td></tr><tr><td rowspan="2">Data set ready</td><td>RSA</td><td>(+) ④</td><td rowspan="2">→ outside</td><td rowspan="2"></td></tr><tr><td>RSB</td><td>(-) ⑰</td></tr><tr><td>DC current</td><td></td><td>⑫ ⑬ ⑭ ⑮ ⑯</td><td></td><td>Do not perform wiring</td></tr><tr><td rowspan="2">Signal ground</td><td>SGA</td><td>⑦ ⑧ ⑲</td><td rowspan="2"></td><td rowspan="2"><div>⑦ ⑧ ⑲ ⑳ ㉑</div>Inside connected equipment</td></tr><tr><td>SGB</td><td>㉑</td></tr><tr><td>Frame ground</td><td>FG</td><td>①</td><td></td><td></td></tr></table> | Signal | Block Diagram | Pin | Signal Direction | Remarks | Send data | SDA | (+) ③ | → outside | | SDB | (-) ⑫ | Receive data | RDA | (+) ② | ← outside | | RDB | (-) ⑮ | Data terminal ready | CSA | (+) ⑤ | → outside | | CSB | (-) ⑱ | Data set ready | RSA | (+) ④ | → outside | | RSB | (-) ⑰ | DC current | | ⑫ ⑬ ⑭ ⑮ ⑯ | | Do not perform wiring | Signal ground | SGA | ⑦ ⑧ ⑲ | | <div>⑦ ⑧ ⑲ ⑳ ㉑</div> Inside connected equipment | SGB | ㉑ | Frame ground | FG | ① | | |
| Signal | Block Diagram | Pin | Signal Direction | Remarks | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Send data | SDA | (+) ③ | → outside | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | SDB | (-) ⑫ | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Receive data | RDA | (+) ② | ← outside | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | RDB | (-) ⑮ | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Data terminal ready | CSA | (+) ⑤ | → outside | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | CSB | (-) ⑱ | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Data set ready | RSA | (+) ④ | → outside | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | RSB | (-) ⑰ | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| DC current | | ⑫ ⑬ ⑭ ⑮ ⑯ | | Do not perform wiring | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Signal ground | SGA | ⑦ ⑧ ⑲ | | <div>⑦ ⑧ ⑲ ⑳ ㉑</div> Inside connected equipment | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | SGB | ㉑ | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Frame ground | FG | ① | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

*Connect pin 21 to the signal ground of the external equipment.

*Connect pin 21 to the signal ground of the external equipment.

POINT

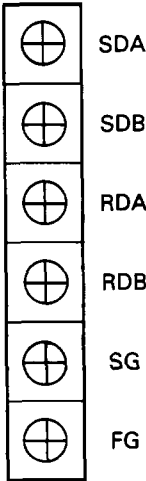
- (1) The maximum transmission speed from the AD51 is 9600 BPS, the maximum receiving speed is 4800 BPS.
- (2) When channel 1 has been set as an I/O console (DIP switch 16 set to ON), the AD51 operating system automatically sets the USART mode:
4800 BPS, even parity, stop bit 1, character data 8

3. SPECIFICATIONS

3.5.2 RS-422 terminal block specifications (CH2)

| Item | Specifications |
|---------------------|--|
| Connected Unit | AD51, personal computer etc. |
| Transmission system | Conforms to EIA. RS-422. |
| Synchronous system | Asynchronous system |
| USART mode setting | <div>Baud rate setting (300, 600, 1200, 2400, 4800, 9600 BPS selectable)</div> <div>Parity bit setting<div>Parity absent</div><div>Parity present<div>Even parity</div><div>Odd parity</div></div></div> <div>Stop bit setting<div>Stop bit 1</div><div>Stop bit 2</div></div> <div>Character data bit setting<div>Data 7 bits</div><div>Data 8 bits</div></div> |

Terminal block pin outs.



| Signal | Block Diagram | Terminal Number | Signal Direction | Remarks |
|--------------------|---------------|-----------------|-------------------|---------|
| Send data (SDA) | | TB1 | → to outside | |
| Send data (SDB) | | TB2 | | |
| Receive data (RDA) | | TB3 | ← from outside | |
| Receive data (RDB) | | TB4 | | |
| Signal ground (SG) | | TB5 | | |
| Frame ground (FG) | | TB6 | | |

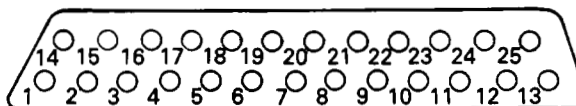
POINT

The maximum transmission baud rate from the AD51 is 9600 BPS. The maximum receiving baud rate is 4800 BPS.

3.5.3 RS-232C connector performance specifications (CH3 and 4)

| Item | Specifications |
|---------------------|---|
| Connected Unit | Console (CH3 only), computer with RS-232C interface, personal computer, printer, modem, etc. |
| Transmission system | Conforms to EIA. RS-232C. |
| Transmission speed | 300, 600, 1200, 2400, 4800, 9600 selectable |
| Synchronous system | Asynchronous system |
| USART mode setting | <ul style="list-style-type: none"> Baud rate setting (300, 600, 1200, 2400, 4800, 9600 BPS selectable) Parity bit setting — Parity absent — Even parity Parity present — Odd parity Stop bit setting — Stop bit 1 Stop bit 2 Character data bit setting — Data 7 bits Data 8 bits Communication control setting — XON/XOFF control Control with DTR terminal <p>*Set CH3 with the front DIP switches (SW1 to 8).</p> |

Connector pin outs.



| Pin Number | Signal Abbreviation | Signal Direction Inside-outside | Description |
|------------|---------------------|---------------------------------|---------------------|
| 1 | FG | | Frame ground |
| 2 | SD | → | Send data |
| 3 | RD | ← | Receive data |
| 4 | RTS | → | Request to send |
| 5 | CTS | ← | Clear to send |
| 6 | DSR | ← | Data set ready |
| 7 | SG | | Signal ground |
| 20 | DTR | → | Data terminal ready |

POINT

- (1) The maximum transmission speed from the AD51 is 9600 BPS, the maximum receiving speed is 4800 BPS.
- (2) When channel 3 has been set as an I/O console (DIP switch 16 set to OFF), the AD51 operating system automatically sets the USART mode:
4800 BPS, parity absent, stop bit 1, character data bit 8

3.6 I/O Interface with Programmable Controller CPU

The digital I/O bus may be used for communication between the PC CPU and the AD51. The following table indicates the function of each signal. The drive number will vary depending on the AD51 slot location; in the table the AD51 is assumed to be in slots 0 and 1 of the main base unit.

- (1) There are 48 input signals to the PC CPU (X00 to X2F) from the AD51.

| Input Number | Description | Address |
|--------------|---|--|
| X00 to X0F | Unused | |
| X10 | Switched on/off by the BASIC program and the contacts used in the sequence program. (Known as General-purpose inputs) | <div><div>Address 6940_H</div><div><div>B7</div><div>B6</div><div>B5</div><div>B4</div><div>B3</div><div>B2</div><div>B1</div><div>B0</div></div></div> <div><div>Address 6920_H</div><div><div>B7</div><div>B6</div><div>B5</div><div>B4</div><div>B3</div><div>B2</div><div>B1</div><div>B0</div></div></div> <div><div>Input number</div><div>X10</div><div>X11</div><div>X12</div><div>X13</div><div>X14</div><div>X15</div><div>X16</div><div>X17</div><div>X18</div><div>X19</div><div>X1A</div><div>X1B</div><div>X1C</div></div> |
| X11 | | |
| X12 | | |
| X13 | | |
| X14 | | |
| X15 | | |
| X16 | | |
| X17 | | |
| X18 | | |
| X19 | | |
| X1A | | |
| X1B | | |
| X1C | | |
| X1D | Switched on to indicate an AD51 CPU fault | |
| X1E | Unused | |
| X1F | | |
| X20 to X2F | Unused | |

(2) There are 48 output signals from the PC CPU to the AD51.

| Output Number | Description | Address |
|---------------|---|---|
| Y00 to Y0F | Unused | |
| Y10 to Y1F | May be used by the PC CPU as extra internal relays (M). | |
| Y20 | Switched on/off in the sequence program and read by the BASIC program (known as General-purpose outputs) | <div> <div>Address 6820_H</div> <div> <div>B7 B6 B5 B4 B3 B2 B1 B0</div> <div> <div>↑</div> <div>↑</div> <div>↑</div> <div>↑</div> <div>↑</div> <div>↑</div> <div>↑</div> <div>↑</div> </div> </div> </div> |
| Y21 | | |
| Y22 | | |
| Y23 | | |
| Y24 | | |
| Y25 | | |
| Y26 | | |
| Y27 | | |
| Y28 | | |
| Y29 | This output may be used to start one task in the AD51 designated as an interrupt program by its task start condition. | <div> <div>Address 6800_H</div> <div> <div>B7 B6 B5 B4 B3 B2 B1 B0</div> <div> <div>↑</div> <div>↑</div> <div>↑</div> <div>↑</div> <div>↑</div> <div>↑</div> <div>↑</div> <div>↑</div> </div> </div> </div> |
| Y2A to Y2F | Unused | |

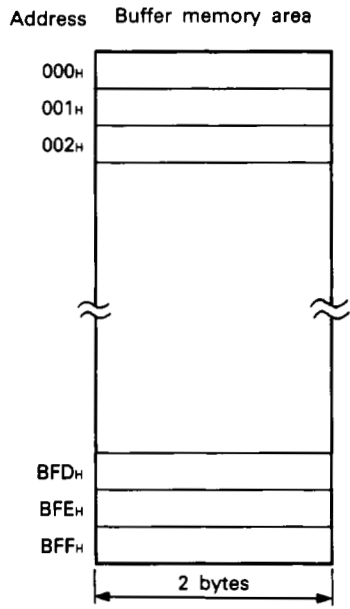
POINT

- (1) Input X1D is switched on when an error is detected by the AD51 hardware. This signal may be used as an interlock in the PC sequence program to control PC accessing of the AD51 buffer memory (i.e. FROM / TO instructions).
- (2) Switching output Y29 on will start the task which has its start condition specified as "interrupt from ACPU". The start condition is defined during "multi task setting".
- (3) Output signals Y2A to Y2F are used by the operating system and must not be switched on or off.

3.7 Buffer Memory

The AD51 uses a buffer memory for data communication with the PC CPU. (The buffer memory is not battery backed.)

- (1) Buffer memory addresses are 000_H to BFF_H (3K words).
See the memory map in Section 3.4.
- (2) Buffer memory data is made up of 16 bits per address.
- (3) The buffer memory is accessed by the AD51 using system subroutines (SR2, SW2). For details, refer to Section 7.2.1.
- (4) The buffer memory is accessed by the PC CPU using the FROM and TO application instructions.
For read and write procedures, refer to Section 7.2.2.
For details of the FROM/TO instructions, refer to the ACPU Programming Manual.



3.8 Communication between AD51 and PC CPU

Any AD51 initiated requests for communication transactions between the AD51 and PC CPU are processed once when the **END**, **FEND** or **COM** instruction is executed by the PC. The time taken to process a system subroutine and the delay times caused by multiple accessing of the PC CPU are explained in this section.

- (1) The following table shows the number of scans taken by the AD51 to process PC transaction subroutines.

| Item | | | System Subroutine | Number of Scans Required for Processing |
|------------------------------|---------------------|-------|-------------------|--|
| Device memory | Batch read | Bit | SADR | 1 scan (2 scans for device "R") |
| | | Word | | |
| | Batch write | Bit | SADW | |
| | | Word | | |
| | Test (random write) | Bit | SADT | |
| | | Word | | |
| | Monitor data entry | Bit | SADM0 | Independent of scan |
| | | Word | | 1 scan for device "R" only (Independent of scan for other devices.) |
| Monitor | Bit | SADM1 | 1 scan | |
| | Word | | | |
| Sequence program | Read | Main | SAAR | 1 scan |
| | | Sub | | |
| | Write | Main | SAAW | 2 scans (1 scan for T/C set value) |
| | | Sub | | |
| Parameter | Read | | SAPR | 2 scans |
| | Batch write | | SAPW | 2 scans |
| | Analysis request | | SAPS | 2 scans |
| Parameter (PC) | Remote run | | SKR | 1 scan |
| | Remote stop | | SKP | |
| | PC type mode | | SPC | |
| Buffer memory | Batch read | | SR2 | Independent of scan |
| | Batch write | | SW2 | |
| Extension file register | Batch read | | SAER | 2 scans |
| | Batch write | | SAEW | |
| | Random write | | SAET | |
| | Monitor data entry | | SAEM0 | 1 scan |
| | Monitor | | SAEM1 | |
| Micro-computer program | Read | Main | SAMR | 1 scan |
| | | Sub | | |
| | Write | Main | SAMW | 2 scans |
| | | Sub | | |
| Comment | Batch read | | SACR | 2 scans |
| | Batch write | | SACW | |
| Special module buffer memory | Batch read | | SATR | 1 scan |
| | Batch write | | SATW | |

POINT

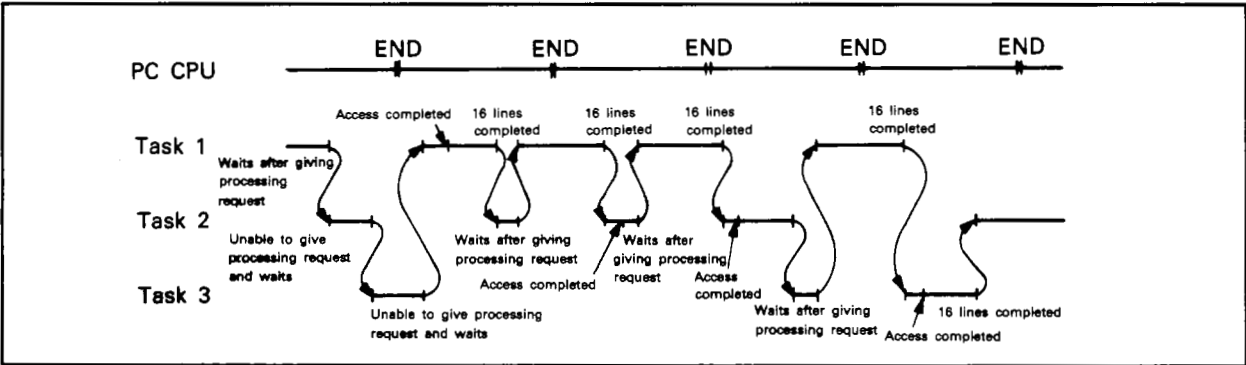
System subroutines SR2 and SW2 (buffer memory read, write) allow max. 3K words (6K bytes) to be transferred between the PC CPU and AD51 at one time independently of the **END**, **FEND** or **COM** instruction execution in the sequence program.

(2) Requests for communication transactions with the PC CPU may also come from other sources, these are listed below and are processed in the same way as AD51 transaction requests. Only one transaction may be processed per PC CPU scan so that a delay of 1 to 5 scans is possible before the AD51 transaction is processed if several of these requests overlap. The following list gives the transaction requests in priority order.

- 1. Programmable controller CPU OS program
- 2. Peripheral equipment (e.g. A6GPP)
- 3. Optical or coaxial data link unit incorporated in CPU module
- 4. Optical or coaxial data link module in 3 hierarchy system AJ71P22/R22
- 5. Processing request from AJ71C24-S3 or second AD51

Hence, if continuous processing requests are received from the A6GPP and AJ71C24-S3, communication between the AD51 and PC CPU is only made once every three scans.

(3) When a system subroutine is called which accesses the PC CPU there is a delay while the PC CPU prepares the appropriate data. During this delay time, the AD51 switches tasks to optimize scan time. In the example shown below, three tasks are executing subroutine which access the PC CPU. Task one provides the first processing request to the PC CPU which prepares the requested data. During this delay the AD51 switches to task 2 which is unable to pass its processing request to the PC which is still dealing with the one from task 1. The AD51 therefore switches to task 3 for which the same situation exists. Only after task 1s request has been fully processed can task 2s request be dealt with. Similarly task 3 must wait until task 2 been processed. For details of other task switching, refer to the GPC-BASIC Handbook.



3.8.1 Transmission time in MELSECNET

Transmission time (T_1) is calculated as follows if data transmission is made to the specified PC CPU which is not used with the AD51 in MELSECNET.

- Master station ↔ local station

Transmission time (T_1) = ((period equivalent to **LRDP** instruction processing time) + (1 scan time of station used with AD51)) $\times 2$

- Master station ↔ remote station

Transmission time (T_1) = ((period equivalent to **RFRP** instruction processing time) + (1 scan time of MELSECNET master station)) $\times 2$

Read "2" marked \sim as "3" when communication is made to the corresponding station for the first time after power on or CPU reset.

Read "2" as "1" from the second communication on when the number of stations communicating is 64 or less.

- Factor of transmission time (T_1) delay

Transmission time obtained from any of the above expressions should be doubled if the command executed requires two scans for transmission (e.g. device "R" write). Transmission time should be multiplied by the (number of stations monitored + 1) if the other link stations are monitored by the A6GPP.

*For more information on data link, see the Data Link System User's Manual.

Example: Reading local station device memory with the AD51 loaded on the MELSECNET master station

(Conditions: $L < LS < M$, $M = 80\text{ms}$, $\alpha 1 = 10\text{ms}$)

$$\text{Transmission time } (T_1) = (M \times 4 + \alpha 1 \times 4 + M) \times 2 \\ = (80 \times 4 + 10 \times 4 + 80) \times 2 = 880$$

where M = MELSECNET master station scan time

$\alpha 1$ = MELSECNET master station link refresh time

LS = link scan time

L = MELSECNET local station scan time

POINT

On some conditions, a considerable delay will occur for data transmission to the PC CPU which is not used with the AD51 in MELSECNET.

Transmission time can be reduced by performing communication only between the AD51 and the PC CPUs used with the AD51 (PC No. FF_H) and using MELSECNET data link (B, W) for communication with the other PC CPUs.

3.9 AD51 Communication

Data communication between the AD51 and external devices (e.g. computer, I/O console, printer) is made using RS-232C and RS-422.

Data communication is made via the transmission and receive buffers which are controlled by the OS of the AD51. The transmission and receive buffers are controlled individually for channels 1 to 4.

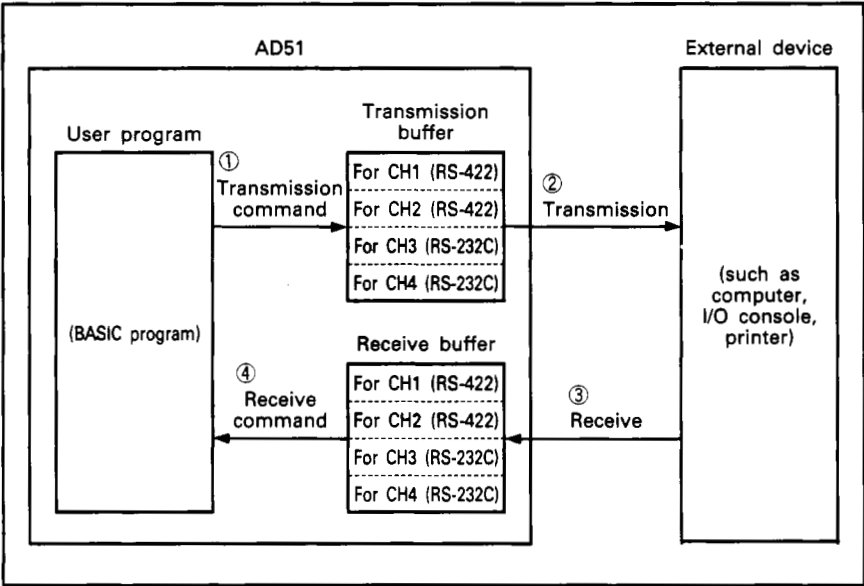
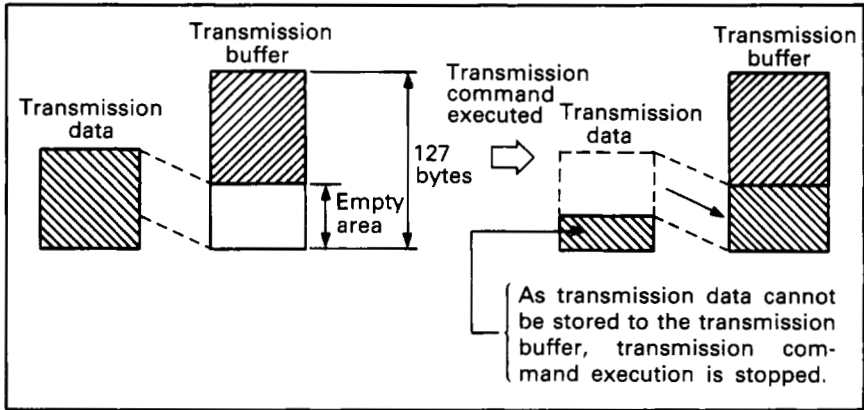
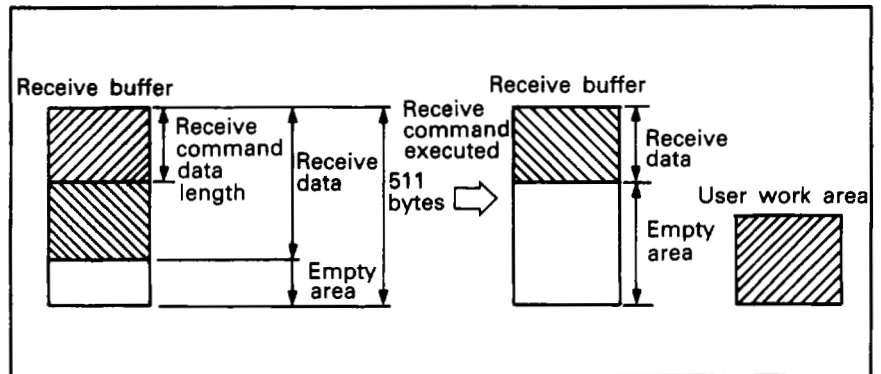


Fig. 3.3 AD51 Communication

- (1) When the transmission command (PRINT, LPRINT, SWB) is executed in the user program, transmission data is stored to the transmission buffer. If the empty area of transmission buffer is less than the transmission data length, the transmission command execution is stopped after as much data is stored to the transmission buffer. The remaining transmission data is stored when an empty area is made in the transmission buffer by data transmission to the external device.



- (2) Data is transmitted in the same order as stored to the transmission buffer when the external device is enabled to receive data by communication control (DTR control or Xon/Xoff control).
- (3) Data received from the external device is stored to the buffer memory.
When the empty area of the receive buffer is reduced, communication control alerts the external device to receive disable.
- (4) Execution of the receive command (INPUT, INKEY, SRB) in the user program transfers data from the receive buffer to the user work area.



3.10 Communication Control

The AD51 has two types of communication control, DTR control and Xon/Xoff control. Communication control defaults to DTR control.
Either control may be selected by executing the corresponding system subroutine as follows:

- DTR control.....Execute SHD.
- Xon/Xoff control.....Execute SHX.

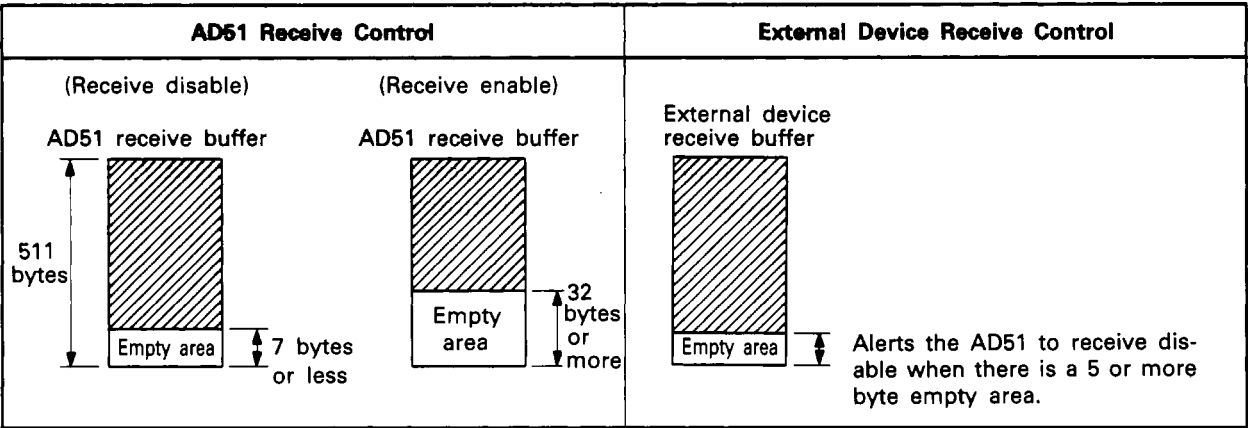
Channel 2 (RS-422 terminal block) is not allowed for communication control change.

(1) AD51 transmission

The AD51 sends data after confirming that the external device is ready to receive.
The external device used must be capable of alerting the AD51 to receive disable when there is an empty area of 5 or more bytes in the receive buffer.

(2) AD51 receive

The AD51 receive buffer has 511 bytes.
The AD51 notifies the external device of receive disable when the empty area of the receive buffer is 7 bytes or less.
The AD51 alerts the external device to receive enable when the empty area of the receive buffer is 32 bytes or more.



POINT

The external device connected to the AD51 must have a receive buffer and be capable of alerting the AD51 to receive disable when there is 5 bytes or more left as an empty area.
Any external device that performs communication handshake in units of 1 byte cannot be connected to the AD51.

3.10.1 DTR control

Performs communication control using the data device ready signal and terminal ready notice signal.

RS-422 : RSA/RSB and CSA/CSB

RS-232C: DSR and DTR

- (a) The data device ready signal is switched on to send data from the transmission buffer to the external device.
The data device ready signal is switched off to stop the transmission.
- (b) The terminal ready notice signal is switched off when the empty area of the receive buffer has become 7 bytes or less.
The terminal ready notice signal is switched on when the receive buffer empty area increases to 32 bytes or more after the receive command (INPUT, INKEY, SRB) is executed in the user program.

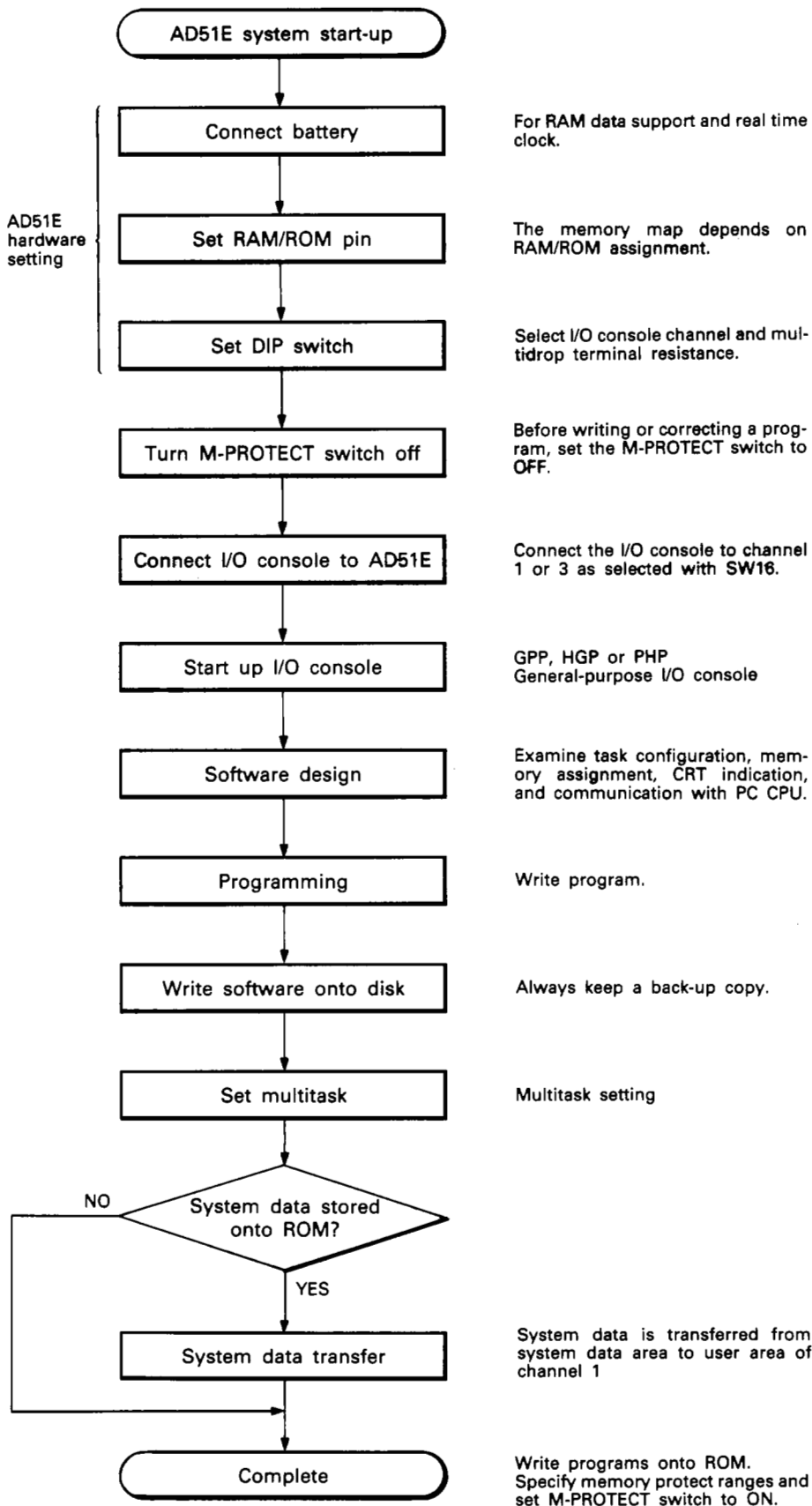
3.10.2 Xon/Xoff control

Performs communication control using the Xon code (11H) and Xoff code (13H).

- (a) Transmission is stopped when the Xoff code is received during transmission of data from the transmission buffer to the external device.
Transmission is resumed when the Xon code is received.
- (b) When the receive buffer empty area is reduced to 7 bytes or less, the AD51 sends the Xoff code to alert the external device to receive disable.
The AD51 sends the Xon code to notify the external device of receive enable when the empty area increases to 32 bytes or more after the receive command (INPUT, INKEY, SRB) is executed in the user program.

4. PRE-START UP PROCEDURES

4.1 General Procedure

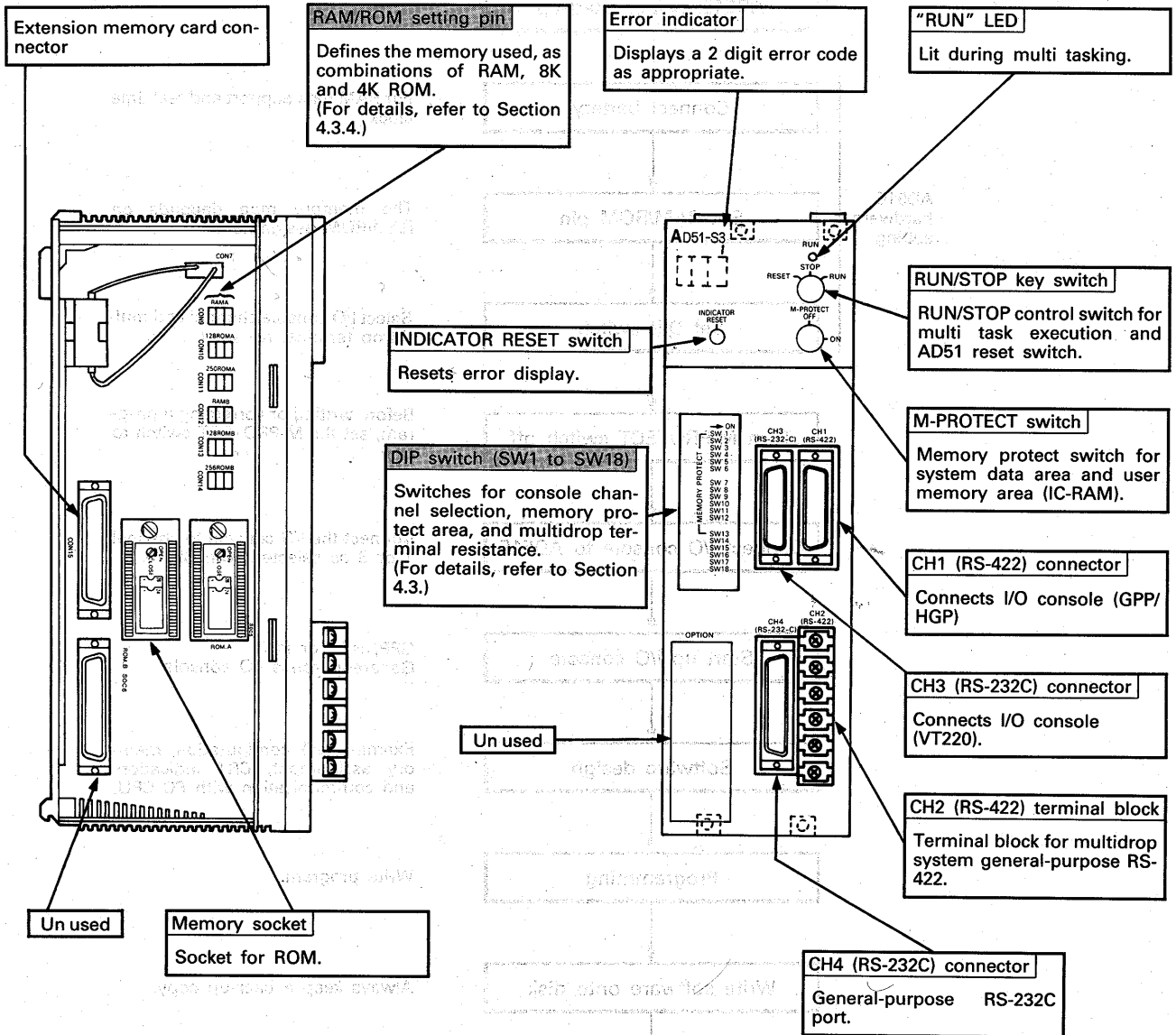


4. PRE-START UP PROCEDURES

MELSEC-A

4.2 Nomenclature

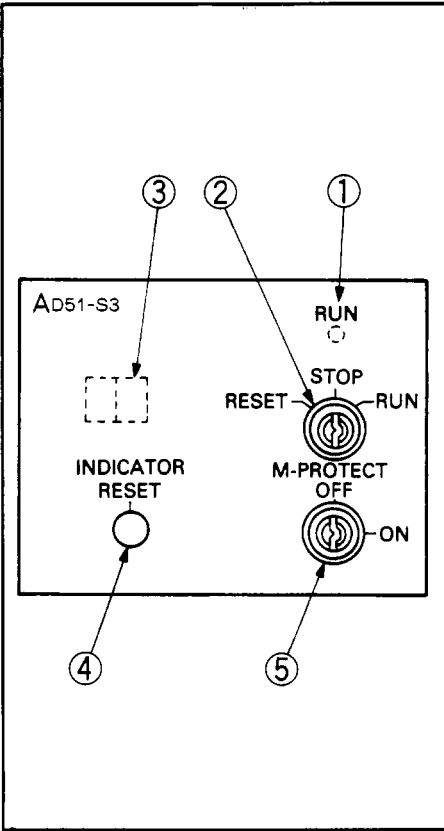
For the setting switches, refer to Section 4.3.



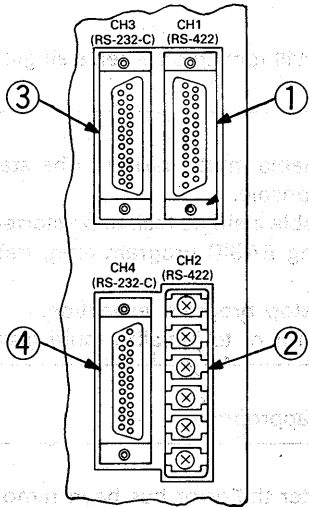
REMARKS

It is necessary to set or load before starting operation.

(1) Switch details

| | | |
|---|---|--|
|  | ① | "RUN" LED On.....During multi tasking Off.....AD51 is not multi tasking. (Will remain off when a single task is RUN.) |
| | ② | "RUN-STOP-RESET" switch RUN/STOP..... <ul style="list-style-type: none">• Set to RUN to enable multi tasking to be started from the input console.• Set to RUN to enable a single task to be started by typing RUN during BASIC programming, debugging etc.• Set to STOP to stop program execution. RESET.....Used to reset an error or to initialize multi tasking. |
| | ③ | Error indicator Displays a 2 digit error code as appropriate. |
| | ④ | "INDICATOR RESET" switch <ul style="list-style-type: none">• Resets the error code display after the error has been removed. The error code will remain if the error has not been cleared.• When several errors have occurred, pressing the reset switch will display consecutive error numbers in the order that they occurred. |
| | ⑤ | "M-PROTECT" switch <ul style="list-style-type: none">• Memory protect for system data (the data entered during multi task setting and BASIC programming) and the user memory.• The memory protect range depends on the DIP switch settings on the front of the unit. For DIP switch details, refer to Section 4.3. |

(2) Connector details



| | |
|---|--|
| ① | CH1 (RS-422) connector Set DIP switch SW16 ON to use this port for programming with the GPP/HGP. Set DIP switch SW16 OFF, to use this connector as a general-purpose port. |
| ② | CH2 (RS-422) terminal block The terminal block is provided for use in the AD51 multidrop system. A maximum of 32 AD51 stations may be included in the multidrop link. This connector may also be used as a general-purpose RS-422 port. |
| ③ | CH3 (RS-232C) connector Set DIP switch SW16 OFF to use this port for programming with the VT-220 or GPP/HGP set DIP switch SW16 ON to use this connector as a general-purpose port. |
| ④ | CH4 (RS-232C) connector General-purpose RS-232C port. |

4.3 Hardware Settings


4.3.1 Memory protect range

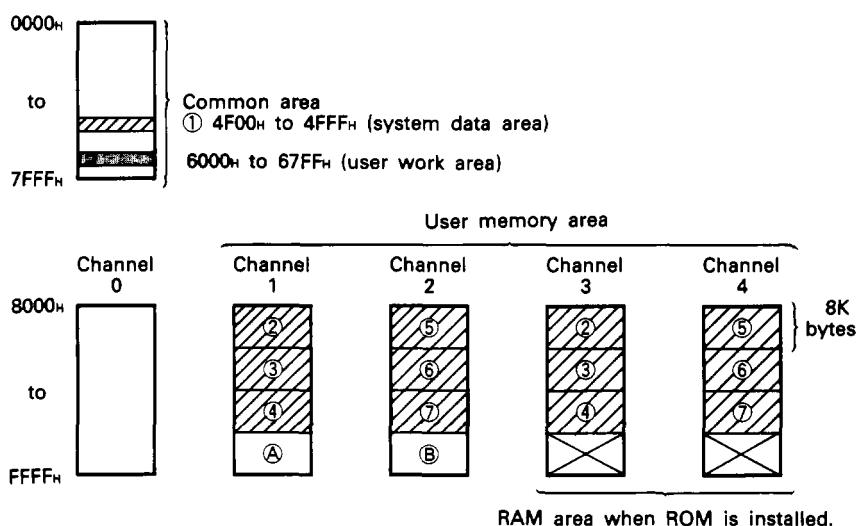
The maximum user program memory capacity is 114K bytes (48K bytes for ROM + 66K bytes for RAM). 48K bytes of the RAM area may be memory protected in units of 8K bytes.

The system data area (for multi task setting data etc.) may also be memory protected.

To set memory protection to the required area, use the DIP switches on the front of the unit as described below:

(1) Memory protect area

Areas marked  in the memory maps below can be memory protected.



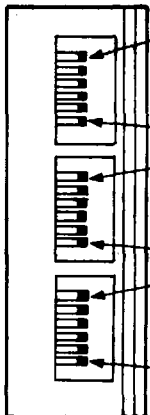
RAM area when ROM is installed.

REMARKS

- (1) The memory protect DIP switch number is shown as ① to ⑦ in the above memory map.
- (2) The memory protect DIP switch numbers for a given RAM address range remain unchanged when ROM is loaded although the channel number has changed from channel 1 to 2 or from 3 to 4.

(2) Memory protect range

Protected RAM address ranges are shown below. The DIP switch is on when the lever points to the right, this is marked on the switch cover.

| Division | DIP Switch Details | Switch Number | Memory Protect Range | | | |
|----------|--|---------------|----------------------|------------|--|--|
| | | | Memory channel | | Memory address | |
| | | | For RAM only | ROM loaded | | |
| ① |  | SW1 | Common area | | 4F00 _H to 4FFF _H | |
| ② | | SW2 | 1 | 3 | 8000 _H to 9FFF _H | |
| ③ | | SW3 | 1 | 3 | A000 _H to BFFF _H | |
| ④ | | SW4 | 1 | 3 | C000 _H to DFFF _H | |
| ⑤ | | SW5 | 2 | 4 | 8000 _H to 9FFF _H | |
| ⑥ | | SW6 | 2 | 4 | A000 _H to BFFF _H | |
| ⑦ | | SW7 | 2 | 4 | C000 _H to DFFF _H | |
| Unused | | | SW8 | Unused | | |
| | | | SW9 | | | |
| | | | SW10 | | | |
| | | | SW11 | | | |
| | | | SW12 | | | |
| | | | SW13 | | | |

POINT

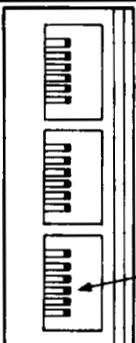
- (1) BASIC program address data and multi task setting data is stored in the system data area 4F00_H to 4FFF_H (256 bytes).
Set memory protect with SW1 after starting multi tasking.
- (2) SW1 must be set to OFF when the system data area data has been stored to ROM.
- (3) Switching the memory protect key to ON protects all areas defined by the DIP switch settings.
- (4) Keep the memory protect switch OFF during BASIC program writing and editing.
- (5) The RAM areas marked ① and ② on the preceeding page and the user work area cannot be memory protected.

4. PRE-START UP PROCEDURES

4.3.2 Console channel

DIP switch SW16 determines which of the two channels, CH1 and CH3 is to be used for the programming console.

When the VT-220 is used SW16 should be switched OFF defining CH3 (RS-232C) as the programming console port. When the GPP/HGP is used the switch is generally switched ON defining CH1 (RS-422) as the programming console port. (RS-232C may also be used)

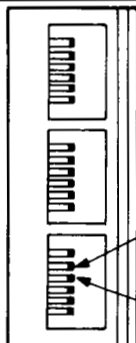
| DIP Switch Details | SW16 Position | CH1 (RS-422) | CH3 (RS-232C) |
|---|---------------|----------------------|----------------------|
|  | ON | GPP/HGP | General-purpose port |
| | OFF | General-purpose port | VT-220 (GPP/HGP) |

POINT

The console setting switch is valid after the AD51 is powered up or reset.
When the console setting has been changed, reset the AD51.

4.3.3 Terminal resistor

A terminal resistor is fitted to prevent distortion of the transmission signal waveform. When a number of AD51s are connected together via an RS-422 link, the two end stations should be set with "terminal resistor present", the remainder with "terminal resistance absent." DIP switches SW14 and SW15 are used to set the terminal resistance as shown below.

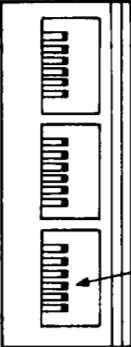
| DIP Switch Details | SW14 Position | SW15 Position | Description |
|---|---------------|---------------|---------------------------|
|  | ON | ON | With terminal resistor |
| | OFF | OFF | Without terminal resistor |

POINT

Both switches should be either on or off.

4.3.4 Setting system data transfer

Set the DIP switch SW17 to ON or OFF as indicated below depending on whether or not the system data is transferred from the ROM of channel 1 to the system data area at power on.

| DIP Switch | SW17 Position | 8000 _H to 8004 _H Data | Description |
|---|---------------|---|---|
|  | ON | Specific pattern | System data is transferred from channel 1 addresses 8000 _H to 80FF _H to system data area addresses 4F00 _H to 4FFF _H . |
| | | Unspecific pattern | Not transferred. |
| | OFF | Specific pattern | Not transferred. |
| | | Unspecific pattern | Not transferred. |

When the system data exists in channel 1 address range 8000_H to 80FF_H, data in 8000_H to 8004_H is in a specific pattern to indicate that the system data exists.

POINT

SW17 must be set to ON when the system data has been stored on ROM and set to OFF when not stored on ROM.

4.3.5 ROM installation

This section describes the installation and settings required for using the ROM.
The ROM sockets should be empty if ROM is not being used.

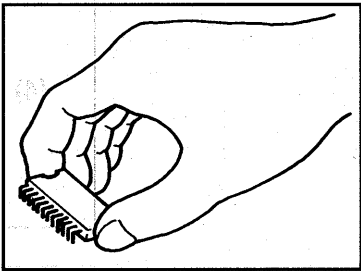
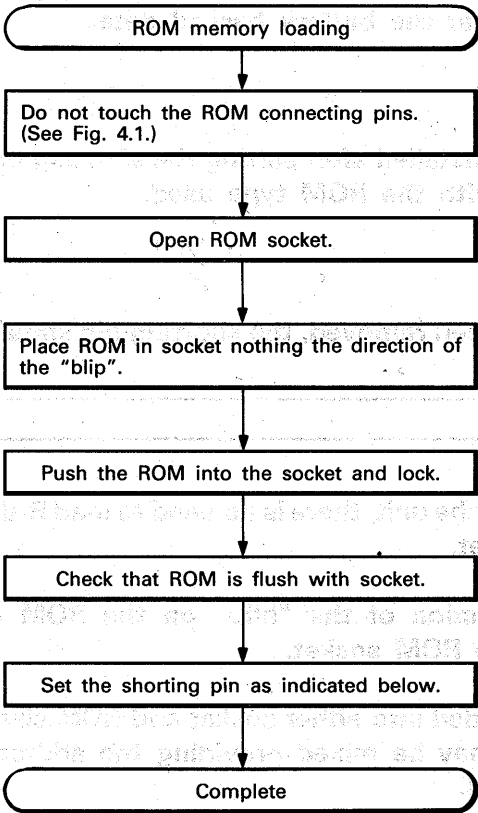


Fig. 4.1 How to Hold ROM

| Shorting pin settings | | |
|---|--|--|
| RAM | ROM (128) | ROM (256) |
| <div>Setting pin</div> <div><div>RAM, A 128ROM, A 256ROM, A RAM, B 128ROM, B 256ROM, B</div><div> RAM A ROM A</div></div> | <div><div>RAM, A 128ROM, A 256ROM, A ROM, B 128ROM, B 256ROM, B</div><div> ROM A</div></div> | <div><div>RAM, A 128ROM, A 256ROM, A RAM, B 128ROM, B 256ROM, B</div><div> ROM A</div></div> |

Fig. 4.2 Pin Settings Depending on Memories

IMPORTANT

- (1) Before ROM is installed, the shorting pin must be set in accordance with the ROM type used.
- (2) After ROM has been installed, setting the shorting pin to RAM may clear the battery backed data.
- (3) ROM installation

ROM should be installed after setting the shorting pin in accordance with the ROM type used.

- (4) ROM removal

After ROM has been removed, the shorting pin should be set to RAM.

POINT

- (1) RAM is built into the unit, there is no need to load RAM into either socket.
- (2) The correct direction of the "blip" on the ROM is indicated on the ROM socket.
- (3) ROM may be loaded into either socket and ROM sizes (27128, 27256) may be mixed providing the address ranges are noted.
- (4) When ROM is installed some RAM address ranges change. (For details, refer to Section 3.4.)
- (5) Cover the EPROM window after it has been programmed.
- (6) Ensure that ROMs are correctly stored and protected.
- (7) Keep the ROM away from static electricity—use anti-static foam where possible.
- (8) The shorting pin is factory-set to RAM (RAM.A and RAM.B connectors).
- (9) Channel 1 shorting pins are marked RAM.A and channel 2, RAM.B.

4.3.6 Loading the battery

The battery is disconnected before leaving the factory to prevent unnecessary battery consumption. The battery plug should be connected to pins CON7 on the circuit board before the AD51 is used. The red wire is positive and the connector is keyed to prevent wrong connection.

5. WIRING

5.1 Wiring Instructions

All AD51 external wiring should be protected against noise.

- (1) Keep cables carrying data at least 100mm(3.94inch) away from main circuit wiring, high voltage cables and PC input and output wiring.
- (2) Ground shield wires or cable shields at one point only.
- (3) Use M4 solderless terminals for connection to the RS-422 terminal block.

5.2 RS-232C Connection

RS-232C connection

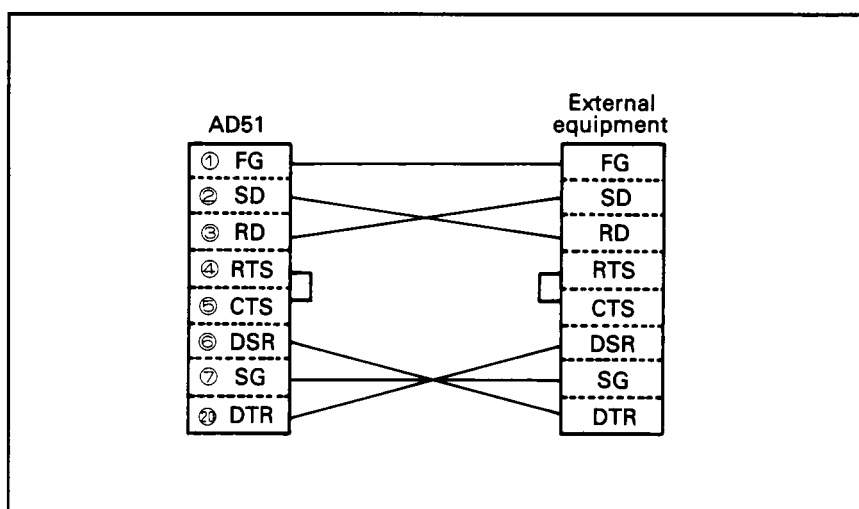


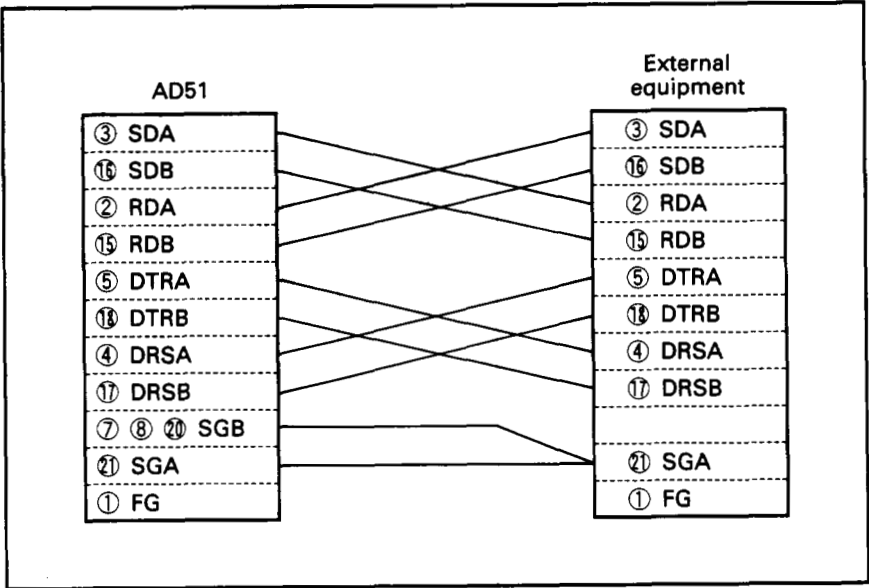
Fig. 5.1 RS-232C Connection Diagram

POINT

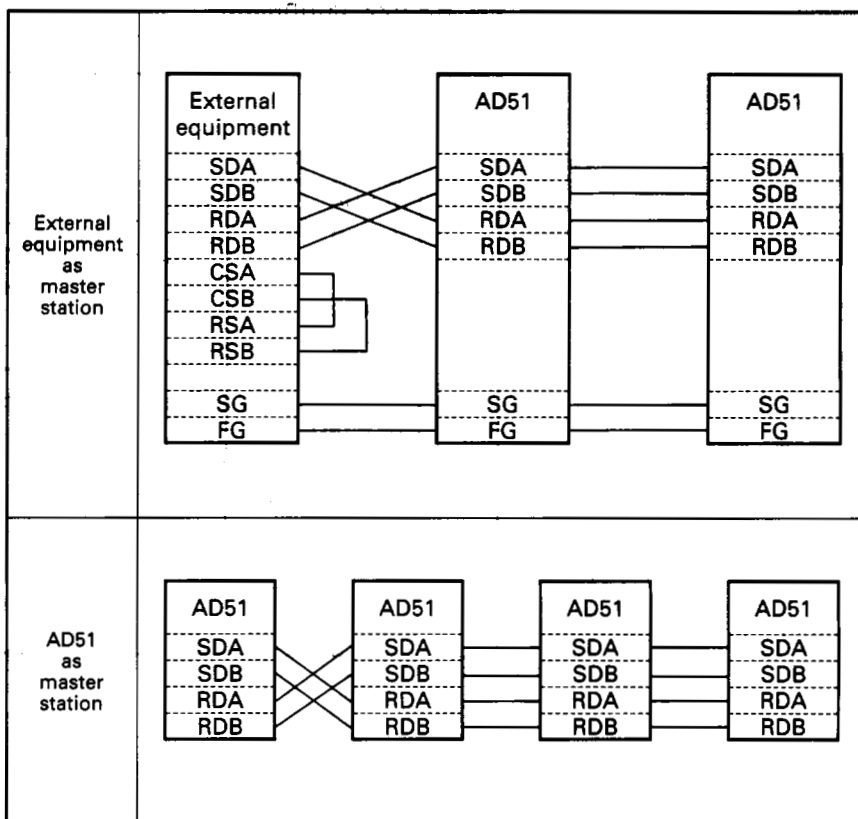
A maximum of 32 stations may be included in the multidrop system with an overall link distance of 500m(547Yd).

5.3 RS-422 Connection

(1) RS-422 connector



(2) RS-422 multidrop connection.



POINT

A maximum of 32 stations may be included in the multidrop system with an overall link distance of 500m(547Yd).

REMARKS

RS-422 multidrop cabling should conform to the following specifications.

| Item | Specifications |
|-----------------------------------|----------------------------|
| Cable type | Shielded cable |
| Conductor resistance (20°C) | 88.0Ω/km or less |
| Insulation resistance | 10,000MΩ·km or less |
| Dielectric strength | 500V DC for 1 minute |
| Electrostatic capacity (1kHz) | 60nF/km or less on average |
| Characteristic impedance (100kHz) | 110 ± 10Ω |

6. AD51 PROGRAMMING NOTES

6.1 BASIC Program Address Data

The following information must be specified before a GPC-BASIC program can be written: program number, program head address, program last address, additional program head address, work area head address, and channel. For further details on setting the data, refer to AD51 Operating Manual.

REMARKS

The following table indicates the function of each of the addresses. Before the BASIC program can be written a BASIC text area must be defined as well as an interpreter work area. The operating system automatically assigns the additional program head address depending on how much of the BASIC text area is vacant.

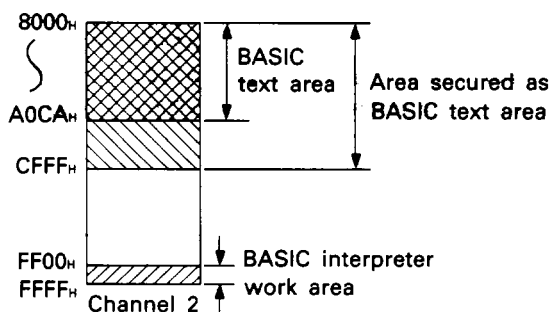
| Item | Description |
|---------------------------------|---|
| Program number | BASIC text number (1 to 8) |
| Program head address | The first address of the BASIC text area (8000 _H onwards) |
| Program last address | The last address of the BASIC text area |
| Additional program head address | Head address of vacant area in BASIC text area. (Automatically set by the O.S.) |
| Work area head address | Work area used for BASIC interpreter. Fixed to 256 bytes. (Not available for user) |
| Channel | Channel for the BASIC text |

- (1) Direct variables (A to Z) are allotted in the BASIC interpreter work area.
- (2) Use the address range D000_H to FFFF_H for @ array variables and indirect variables. The hatched areas in the example below may not be used.

Example:

Channel 2 (8000_H to FFFF_H) data has been set as follows:

1. Program head address 8000_H
2. Program last address CFFF_H
3. Additional program head address A0CB_H
4. Work area head address FF00_H
5. Channel 2

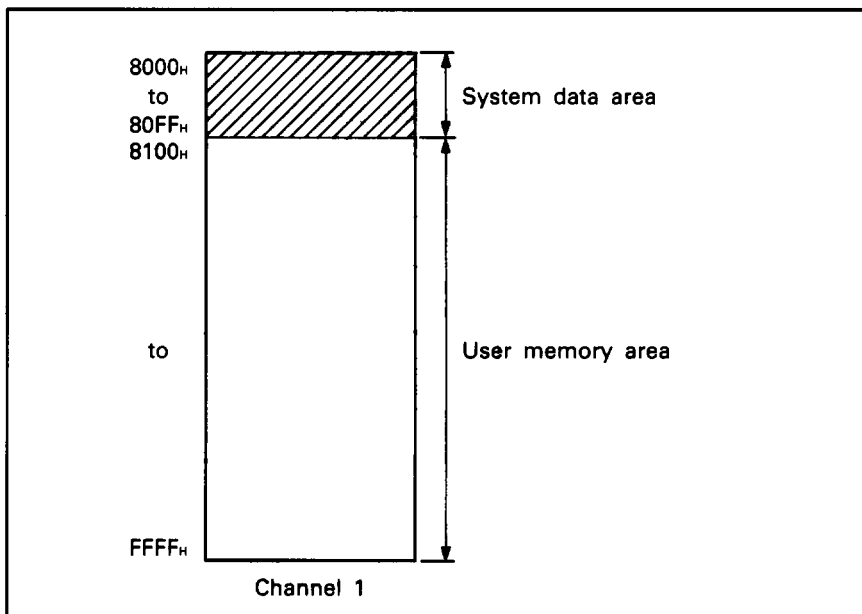


- (3) The work area must come after the text area and the work area head address must have 00 in the two least significant digits. 256 bytes are used for the BASIC interpreter work area starting at the work area head address.
- (4) When two or more tasks are written in the same channel make sure that the program areas and BASIC interpreter work areas do not overlap each other.
 - a) Program data will be corrupted in overlapping memory areas.
 - b) Multitasking results will be invalid if the BASIC interpreter work area for a given task is overlapped by program data from a different task. Independent running of that task however is valid.

| | BASIC Program Addresses | TASK 1 | TASK 2 | Memory Map |
|-----------------------------------|--|--|--|--|
| Correct example | Task Program head address Program last address Additional program head address Work area head address Channel | 1 8000H AFFFH A74CH F000H 1 | 2 B000H DFFFH C851H F800H 1 | <p>8000H B000H DFFFH F000H F800H FFFFH</p> <p>BASIC text area (task 1) BASIC text area (task 2) BASIC interpreter work area (task 1) BASIC interpreter work area (task 2)</p> |
| Overlapping program area | Task Program head address Program last address Additional program head address Work area head address Channel | 1 8000H AFFFH A74CH F000H 1 | 2 A000H DFFFH B851H F800H 1 | <p>8000H B000H BFFFH DFFFH F000H F800H FFFFH</p> <p>BASIC text area (task 1) Overlapping area BASIC text area (task 2) BASIC interpreter work area (task 1) BASIC interpreter work area (task 2)</p> |
| Overlapping interpreter work area | Task Program head address Program last address Additional program head address Work area head address Channel | 1 8000H AFFFH A74CH F000H 1 | 2 B000H DFFFH C851H F000H 1 | <p>8000H B000H DFFFH F000H FFFFH</p> <p>BASIC text area (task 1) BASIC text area (task 2) BASIC interpreter work area (task 1) BASIC interpreter work area (task 2)</p> |

- (5) When the system data is stored onto ROM, the program head address must be 8100_H or a subsequent address because addresses 8000_H to 80FF_H of channel 1 are used as a system data area.

Setting the program head address between 8000_H and 80FF_H will corrupt the BASIC program and disable normal program execution.



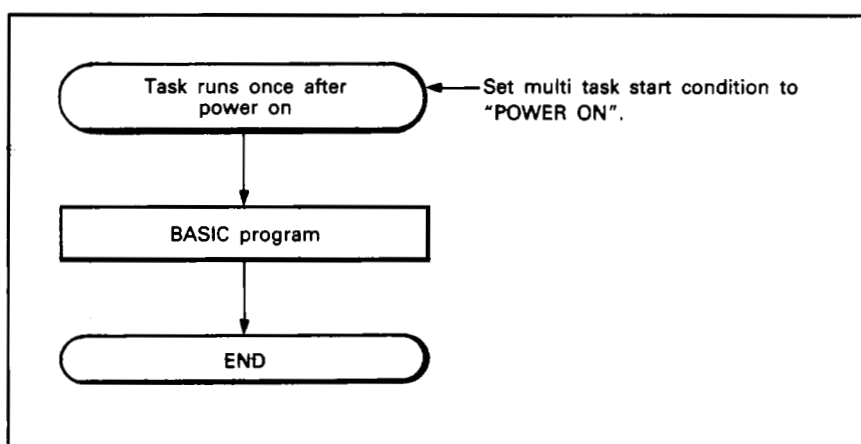
6.2 Start Conditions

There are 4 types of BASIC program execution formats:

- (1) Program runs once after power on.
- (2) Program runs continuously after power on.
- (3) Program runs after an interrupt signal from the PC CPU.
- (4) Program runs at preset intervals in real time.

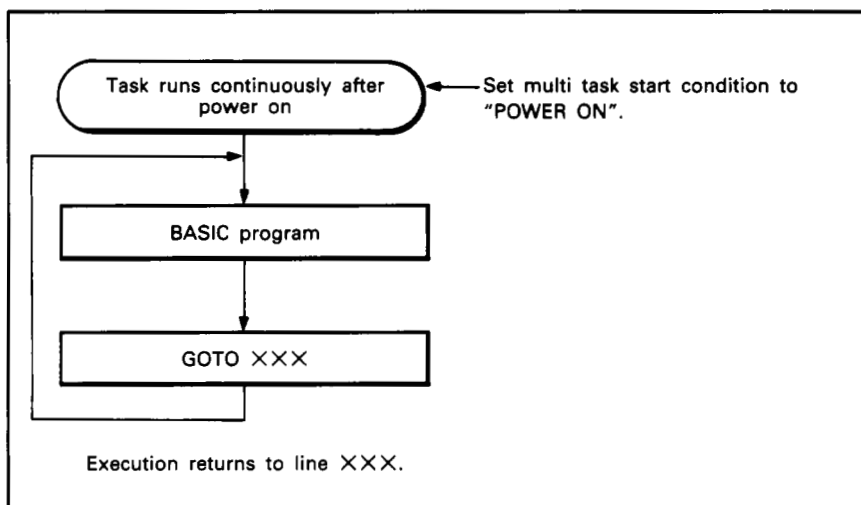
6.2.1 Program runs once after power on

Write the BASIC program so that "END" is executed as the final instruction and set the task start condition to "POWER ON".



6.2.2 Program runs continuously after power on

Write the BASIC program using the "GOTO" command to continue program execution and set the task start condition to "POWER ON".

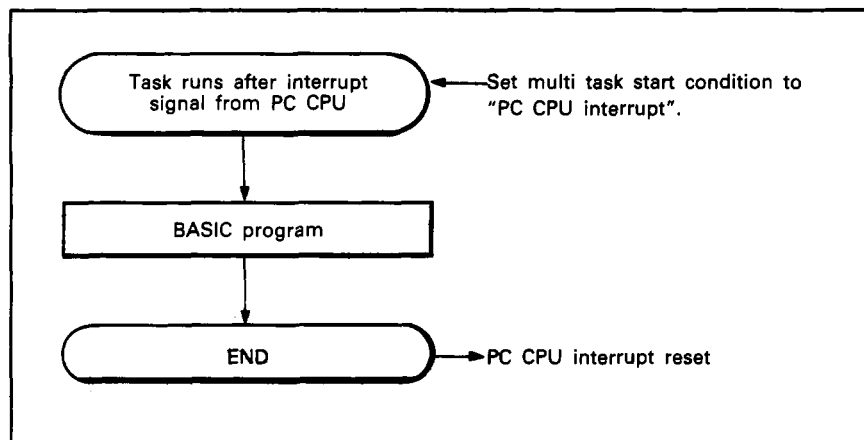


6.2.3 Program runs after an interrupt signal from the PC CPU

Set the task start condition to "CPU INT" (CPU interrupt). The program is then run when the rising edge of the AD51 interrupt signal is received from the PC CPU.

For programming information see Section 7.4.

- (1) Write the BASIC program so that "END" is executed as the final instruction. When "END" is executed the AD51 interrupt condition is reset. The interrupt program will not run again until the rising edge of the AD51 interrupt signal is received from the PC CPU.



- (2) Only one task may be defined as PC interrupt start. More than one will generally lead to "ORST" error.

6.2.4 Program runs at preset intervals in real time

Select starting condition "REAL TIME INT" (real time interrupt).

The real time interrupt interval (i.e. the time between interrupt signals) should be longer than the total time taken for the interrupt program to reach the END instruction including the time required by other tasks. An "ORST" error is detected if a second real time interrupt signal is given before the interrupt program has executed its END command.

POINT

For tasks other than the one started by the interrupt signal from the PC CPU (Section 6.2.3), any system subroutine must not be executed to access the PC CPU within five seconds after the PC CPU is run.

A PC down error will occur if the system subroutine is executed within five seconds.

6.3 Notes on the Use of BASIC Commands

6.3.1 Key input commands

Key inputs (INPUT and INKEY commands) to the AD51 via one channel (as specified by the ZIDV command) should only be made to one task.

Since tasks are executed in order of task numbers, any data keyed in to a task can only be read at certain intervals. If a key is pressed and more than one task is waiting for data from the specified channel, only the first task to execute the INKEY or INPUT instruction will read the key input.

The other tasks will then continue waiting until a key is pressed while they are being run.

| Task 1 | Task 2 |
|------------------|------------------|
| 100 REM "TASK 1" | 100 REM "TASK 2" |
| to | to |
| 200 ZIDV 1 | 500 ZIDV 1 |
| 210 A=INKEY | 510 B=INKEY |
| to | to |

Example: When both task 1 and task 2 are waiting for key input from channel 1, pressing a key will only write data to one of variables A or B.

6.3.2 Printing commands

The printing commands are "PRINT" and "LPRINT".

(1) Difference between PRINT and LPRINT commands

[PRINT]

Used when the printer is connected to the console channel (channel 1 or 3 set with DIP switch 16) or the channel specified with ZODV.

[LPRINT]

Used when the printer is connected to the channel specified in the printer setting.

(2) Sharing of a single printer between tasks.

When several tasks are sharing the use of a printer ensure that interlock flags are provided in the work area to prevent two or more tasks attempting to access the printer simultaneously.

- (3) Note that with printers that use the CR code (0D_H) to initiate printing (K6PRE, K7PRE etc.), writing a comma (,) after the statement in the PRINT or LPRINT command, stops the AD51 from sending the CR code.
Printing is therefore not initiated.

(4) Notes on the use of the KD51PR

The KD51PR may be connected to either of the two AD51 RS-232C ports. When using the KD51PR note the following:

- (a) The KD51PR will print "?" if data is sent from the AD51 while it is printing or during paper feed. This may be avoided by using the program shown in Example 2.
Example 1 shows a program which repeatedly prints the letters "ABCDE" and the resultant KD51PR print out.

[Example 1]

Use of KD51PR in "2K buffer OFF, buffer full set" mode

BASIC program

```
100 LPRINT "ABCDE"
110 LPRINT * $ 03,
120 GOTO 100
```

--Print data and control codes (CR code, LF code) are sent from AD51.

--Print control code 03_H is sent from AD51.

--Upon execution of line 120, execution returns to line 100. Data is continuously written to the KD51PR while it is still printing.

Print result

```
ABCDE
?A?B?D?
?
?E?
?E??D?
?D?E??A?C?E?
```

[Example 2]

KD51PR setting 2K buffer ON, buffer full set

BASIC program

```

100 LPRINT "ABCDE"
110 LPRINT * $ 03,
120 ZTIME 400
130 GOTO 100

```

--Print data and control codes (CR code, LF code) are sent from AD51.

--Print control code 03H is sent from AD51.

The KD51PR starts printing after it receives the 03H code. The ZTIME instruction allows a time delay before the next set of data is sent.

Print result

```

ABCDE
ABCDE
ABCDE

```

In this case, it takes 4 seconds after 1-time print command to print termination.
 ("ABCDE" print 2 seconds)
 (CR, LF (line feed) 2 seconds)

- (b) When the 2K buffer is set to OFF in the KD51PR, any string of characters sent which is more than one line long will lead to an overrun error when the LF code is given. (The receive buffer is 32 characters long). See below:

```

100 LPRINT "ABCDEFGHIJKLMNOPQRSTUVWXYZABCDEF"
           <32 characters>

```

To overcome this, write a comma (,) after the PRINT statement to stop the LF code from being given as below:

```

100 LPRINT "ABCDEFGHIJKLMNOPQRSTUVWXYZABCDEF,"

```

6.3.3 CRT display commands

Any commands addressed to a CRT on a given channel should come from one task only.

There is no management of display commands between tasks.

[CRT display commands]

CLS, ZCON, ZCOFF, ZNOR, ZCRV, PRINT, LOCATE

POINT

Any program controlling the display on one CRT should be written in one task only.

6.3.4 OPEN and CLOSE commands

Note the following precautions when using the OPEN and CLOSE commands.

- (1) The channel specified as that for the console by DIP switch SW16 and the channel selected for the printer on the printer setting screen are automatically opened by the AD51 OS. The communication mode for other channels must be set in the user program using the OPEN command. These two channels default to closed.
- (2) The OPEN command is used to specify the communication control at the RS-232C or RS-422 port. It also initializes the AD51 send and receive buffers. Executing this command therefore re-defines the communication mode for the specified port and clears both AD51 buffers at that port.
- (3) The CLOSE command initializes the AD51 communication control and buffers in the same way as the OPEN command however in this case the port is left in a read/write disable state.

POINT

Before using the CLOSE command, ensure that the transmit buffer is empty by using subroutine "STC". Data will be lost if the buffer is initialized while it still contains data.

- (4) OPEN and CLOSE commands are common to all tasks. Therefore, any channel opened by one task can be used in subsequent tasks without having to repeat the OPEN command.

6.3.5 Z commands

The ZMOV, ZRD1, ZRD2, ZWR1 and ZWR2 commands are not available for AD51, PC CPU transactions. Access to the PC CPU data is via system subroutines. For details refer to Section 3.2.2.

6.4 Transmission Commands to External Device

The PRINT, LPRINT and SWB commands are used to transmit data from AD51's RS-422 or RS-232C to the external device.

POINT

**Any transmission command is complete when transmission data is stored to the transmission buffer.
Use system subroutine STC to check that the transmission data has been sent to the external device.**

6.4.1 PRINT command

- (1) The channel used is determined by the console channel (set with DIP switch 16) or ZODV command.
- (2) Data transmitted depends on the PRINT command designation form as described below:

| Designation | Processing |
|--------------------------------|---|
| Expression (variable) | Value of expression (variable) is converted into a 6-digit decimal ASCII code and transmitted to the external device. |
| \$ expression (variable) | Value of expression (variable) is converted into a 4-digit hexadecimal ASCII code and transmitted to the external device. |
| ? expression (variable) | Value of expression (variable) is converted into a 2-digit hexadecimal ASCII code and transmitted to the external device. |
| . expression (variable) | Value of expression (variable) is converted into a real ASCII code and transmitted to the external device. |
| # expression (variable) | Value of expression (variable) is converted into a decimal ASCII code of specified digits and transmitted to the external device. |
| * expression (variable) | Value of expression (variable) is regarded as an ASCII code and transmitted unchanged to the external device. |
| (Character string variable) | Data stored in character string variable is regarded as an ASCII code and transmitted unchanged to the external device. |

- (3) The PRINT command may be used with ASCII codes 00_H to FF_H. Note that 00_H and 0D_H are transmitted as indicated below:

When transmission data is 00_H, 0D_H is sent.

When transmission data is 0D_H, 0D_H and 0A_H are sent.

POINT

The following program should be written when sending 0D_H or 0D_H and 0A_H by the PRINT command.

0D_H: PRINT *0, 0D_H, 0A_H: PRINT *\$D

6.4.2 LPRINT command

- (1) Transmits data from the channel specified on the printer setting screen.
The LPRINT and LLIST commands cannot be used if NOTHING has been selected for the printer type.
- (2) Data transmitted in accordance with the LPRINT command designation is as indicated in Section 6.4.1.
- (3) Data transmitted by the LPRINT command is as indicated in Section 6.4.1.
- (4) The LPRINT command is not controlled on a task basis.

6.4.3 System subroutine SWB

- (1) Transmits data of the specified length from the specified channel.
- (2) The transmission time can be set in increments of 10ms.
A time-out occurs if transmission is not completed (data is not stored to the transmission buffer) within the transmission period.
The error status and untransmitted data byte length can be read.
- (3) Transmission data is stored to the AD51's common area 6000_H to 67FF_H.
Data in 00_H to FF_H is regarded as an ASCII code and sent unchanged.

6.5 AD51 and PC CPU Reset

The following explains the effects of resetting the AD51 and the PC CPU.

(1) AD51 reset operation

- 1) The AD51 processes its programs as though the power has been switched on.

| | |
|-----------------------------------|---|
| With multi task start already set | Executes multi task. |
| Without multi task start | Displays the mode select menu on the console. |

- 2) All the AD51 general-purpose inputs are switched off.
- 3) During reset, there is no accessing of the PC CPU.
- 4) There is no signal by which the PC CPU can know that the AD51 has been reset.
- 5) Any **FROM** or **TO** instructions executed by the PC CPU when the AD51 is reset will be invalidated.

(2) PC CPU reset operation

- 1) All the AD51 general-purpose outputs are switched off.
- 2) Resetting the PC CPU disables access by the AD51 to the PC CPU for about five seconds after the PC CPU is set to RUN. If the PC CPU is accessed during this period, "PC DOWN ERROR" or "TIME OUT ERROR" is detected.

6.6 Notes on BASIC Programming

- (1) Before making additions or corrections to the BASIC program or changing the program to its final format with the RUN or COMPILE command, set the "M-PROTECT" switch to OFF.
- (2) Before executing multi task, remove all STOP and BREAK commands.
- (3) Always RUN or COMPILE the BASIC program after it has been completed.
The RUN or COMPILE commands change the program into a format suitable for multitasking. If the program is not formatted the CPU may misoperate. Remember to RUN or COMPILE programs before writing them to ROM.

7. COMMUNICATION WITH PROGRAMMABLE CONTROLLER CPU

The AD51 occupies 48 I/O points and is provided with 13 digital inputs and 10 digital outputs as well as a 3K word buffer memory. The following section describes communication between the PC CPU and the AD51. It is assumed that the AD51 is located at slot 0 and 1 of the main base unit and that any BASIC program is written in channel 1.

7.1 General-Purpose I/O Read/Write

The AD51 can access the general-purpose I/O immediately and continuously using the "PEEK" and "POKE" commands. This eliminates the need to use the data memory read/write system subroutines which are only processed when the "END" or "COM" instruction is executed in the PC CPU.

Output Y29 from the PC CPU can be used as an interrupt signal to the AD51, see Section 7.9.

7.1.1 General-purpose I/O addresses

The general-purpose I/O assigned to the AD51 are accessed via the following addresses in the user work area:

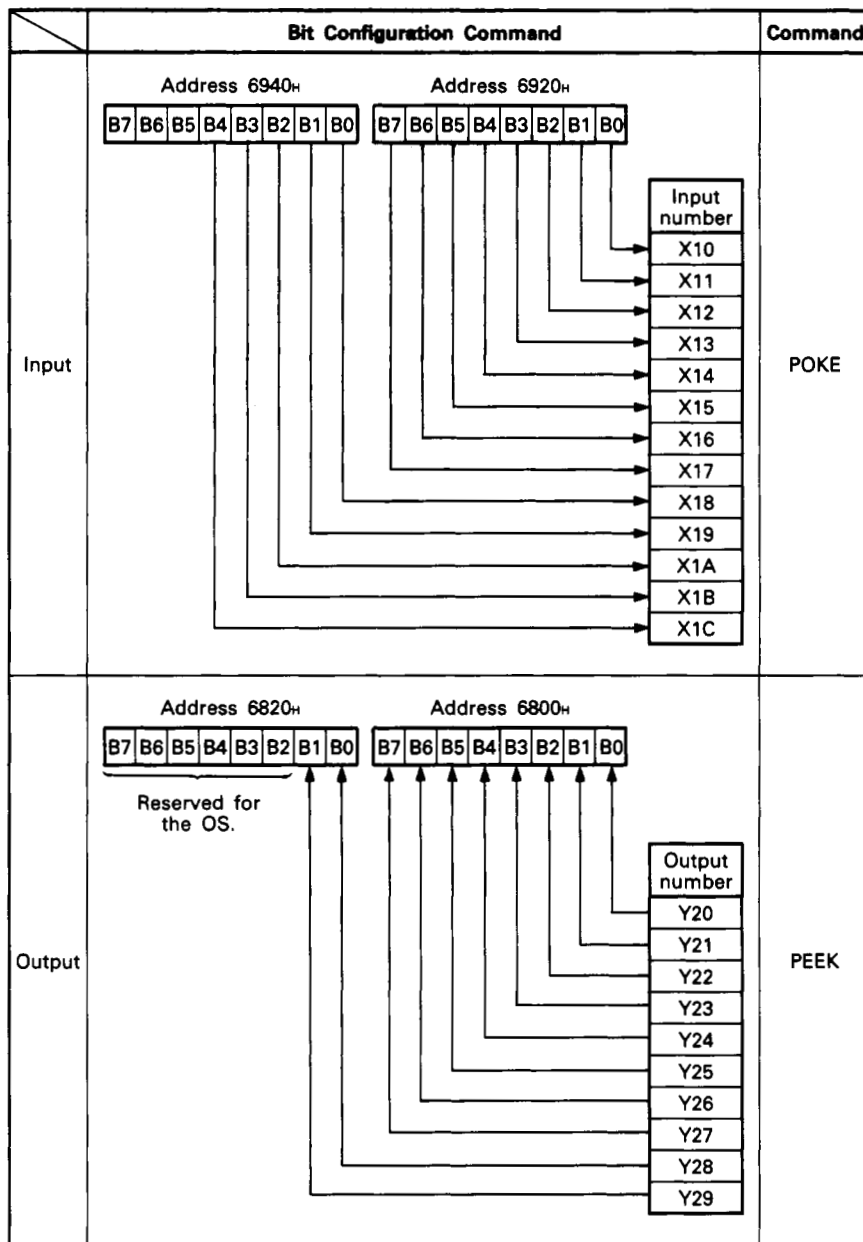
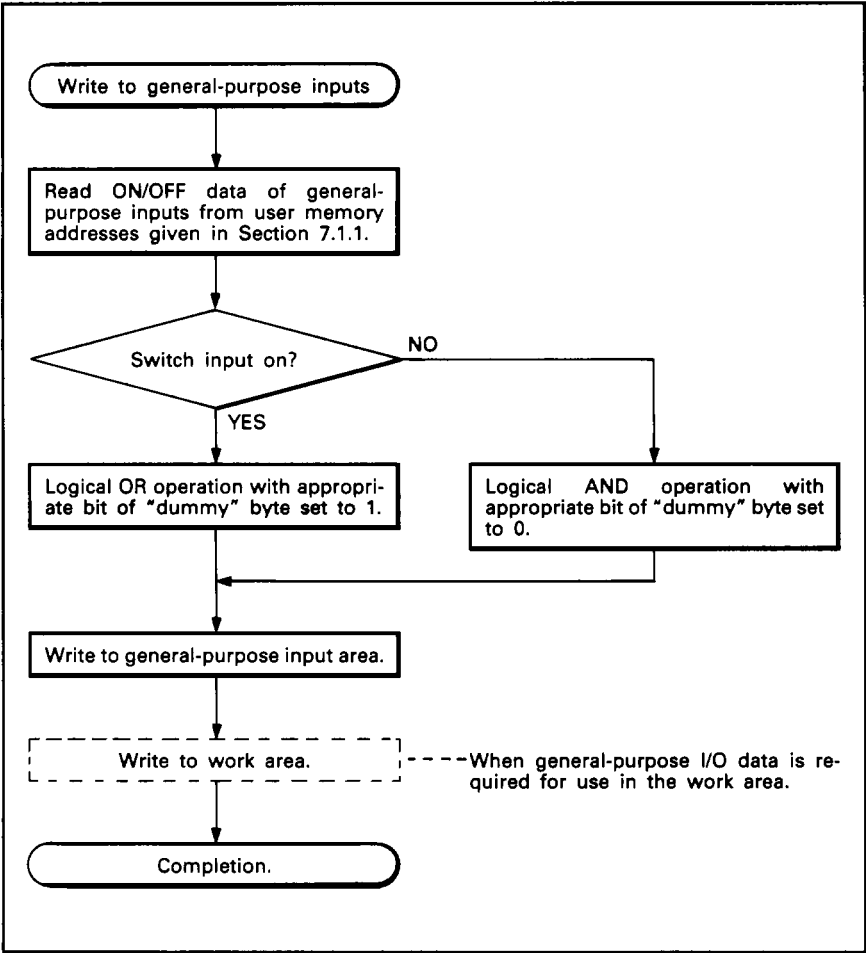


Table 7.1 General-Purpose I/O Data Configuration

7.1.2 Write to general-purpose input

To wrote to a specified general-purpose input the appropriate bit in the relevant byte of memory must be set or reset. The memory addresses for general-purpose inputs and outputs are given in Section 7.1.1.

This section describes the direct accessing of general-purpose I/O data using the PEEK and POKE commands. The system subroutine "SADR" may also be used to access this data but system subroutines are only processed after the "END" or "COM" instruction has been executed in the PC CPU.



[Program example]

Inputs X13 and X1B are turned on and off respectively and the resultant on/off condition is stored in head addresses 6000_H and 6001_H.

(1) Program to switch on X13

100 A=PEEK(\$ 6000)..... Reads address 6000_H from the common area (This byte is all '0's).
 110 B=A/\$ 08 Executes a logical OR operation with address 6000_H and a "dummy" byte containing the binary representation of 8 as follow:

| | Bit pattern |
|---------------------------|-------------|
| Address 6000 _H | 00000000 |
| 8 ₂ | 00001000 |
| OR result | 00001000 |

The result is written to variable B.

120 POKE \$ 6920, B Variable B is written to address 6920_H (i.e. the user work area reserved for accessing the general-purpose I/O).
 130 POKE \$ 6000, B The result is also written to address 6000_H.
 140 END

(2) Program to switch off X1B

100 A=PEEK(\$ 6001)..... Reads address 6001_H from the common area (This byte is all '0's').
 110 B=A&\$ F7 Executes a logical AND operation with address 6000_H and a dummy byte containing the binary representation of F7 as follows:

| | Bit pattern |
|---------------------------|-------------|
| Address 6000 _H | 00000000 |
| F7 ₂ | 11110111 |
| AND result | 00000000 |

The result is written into variable B.

120 POKE \$ 6940, B Variable B is written to address 6940_H (i.e. the user work area reserved for accessing the general-purpose I/O).
 130 POKE \$ 6001, B The result is also written to address 6001_H.
 140 END

POINT

*: X1D to 1F cannot be turned on/off in the BASIC program. The value marked * in line 110 of (2) may also be 17_H (bits 5 to 7 = 0). If AND operation is performed, the same result is obtained.

7.1.3 Read from general-purpose output

A similar procedure may be used to read the general-purpose outputs using the logical AND operation. The work area addresses are given in Section 7.1.1. Note that bits 2 to 7 of address 6820_H are reserved for the operating system and should be masked.

[Program example]

Program for ON/OFF monitor of Y20 to 27

```
100 A=PEEK($ 6800)..... Reads address 6800H to variable A.
110 B=1 ..... Variable B set to 1.
120 FOR I=0 TO 7
130 IF (A&B)#0 PRINT "ON" ;
      GOTO 150 ..... Performs AND operation of variable A
                        and variable B, and if relevant bit is on,
                        displays "ON".
140 PRINT "OFF"
150 B=B*2 ..... Shifts the contents of variable B one bit
                        to the left.
160 NEXT I
170 END
```


7.2 Read/Write of Buffer Memory

The buffer memory can be accessed by both the BASIC program and the sequence program.

A maximum of 3K words can be transferred with one instruction.

7.2.1 Read/write with BASIC program

The following system subroutines are used to access the buffer memory.

For details refer to the GPC-BASIC Supplementary Handbook.

| | |
|-------------------------|-----|
| Read from buffer memory | SR2 |
| Write to buffer memory | SW2 |
| Read/write retry time | SC2 |

REMARKS

The buffer memory cannot be accessed by the AD51 if the PC CPU is already executing a **FROM** or **TO** instruction. In this case the AD51 will retry communication according to its retry time setting.

The retry time is set at 10ms unless changed using the SC2 system subroutine.

[Program example]

- (1) Program to read 256 words from buffer memory head address 200_H to AD51 memory head address F000_H.

```

100 A= $ E000 ..... Indirect variable head address
110 A(0)= $ 200 ..... Data source buffer memory head
                        address
120 A(1)= $ F000 ..... Data destination head address
130 A(2)= $ 100 ..... Number of words
140 B=CALL(0, $ 8000, 1, A) ..... System subroutine SR2 reads buffer
                        memory.
150 IF B#0 PRINT "ERROR",
    B; GOTO 140 ..... Checks for errors in SR2 execution.
160 END

```

- (2) Program to write data from work area addresses E700_H to E77F_H in the BASIC program to buffer memory addresses 350_H to 38F_H.

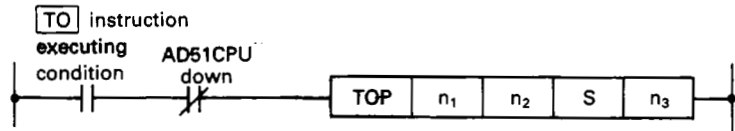
```

100 C= $ E100 ..... Indirect variable head address
110 C(0)= $ 350 ..... Data destination buffer memory head
                        address
120 C(1)= $ E700 ..... Data source buffer memory head
                        address
130 C(2)= $ 40 ..... Number of data words
140 D=CALL(0, $ 8003, 1, C) ..... System subroutine SW2 writes data to
                        buffer memory.
150 IF D#0 PRINT "ERROR",
    D; GOTO 140 ..... Checks for errors in SW2 execution.
160 END

```


(2) Write to buffer memory.....**TO**, **TOP**, **DTO**, **DTOP** instructions

Format



| Symbol | Description | Usable Devices |
|----------------|--|---------------------|
| n ₁ | 16th I/O address of the AD51 omitting the least significant digit. | K, H |
| n ₂ | Buffer memory head address of data destination. | K, H |
| S | Head device number of data source | T, C, D, W, R, K, H |
| n ₃ | Number of words of data to be written | K, H |

[Program example]

Fig. 7.4 shows a program to write data from PC data registers D200 to D263 to buffer addresses 100_H to 13F_H when X1F turns on with the system configuration shown in Fig. 7.2.

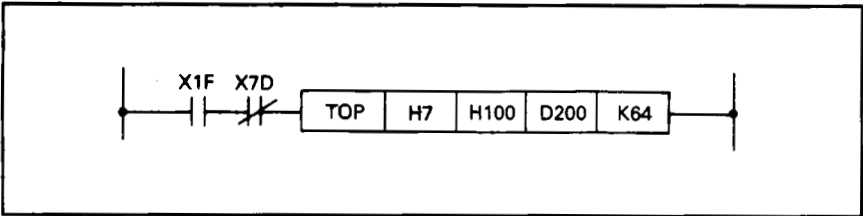


Fig. 7.4 Buffer Memory Write

7.3 Device Memory Read/Write

This section explains how the AD51 communicates with the PC CPU device memory to read present value data and to write new values.

7.3.1 System subroutines and device ranges

The following system subroutines are used by the AD51 to access the PC CPU device memory.

(1) System subroutine types and functions

| Item | | | System Subroutine | Processing | Number of points processed per PC CPU, AD51 transaction. | PC CPU State | |
|---------------|----------------------------|------|-------------------|--|--|--------------|------------|
| | | | | | | During STOP | During RUN |
| Device memory | Batch read | Bit | SADR | Reads data from bit device (such as Y and M) (for 1 point). | 256 points | ○ | ○ |
| | | Word | | Reads data from bit devices (such as Y and M) (for 16 points). | 32 words (512 points) | | |
| | | | | Reads data from word device (such as D and R) (for 1 point). | 64 points | | |
| | Batch write | Bit | SADW | Write data to bit device (such as Y and M) (for 1 point). | 160 points | ○ | ○ |
| | | Word | | Writes data to bit devices (such as Y and M) (for 16 points). | 10 words (160 points) | | |
| | | | | Writes data to word device (such as D and R) (for 1 point). | 64 points | | |
| | Test (During random write) | Bit | SADT | Sets/resets any specified bit device (such as Y and M) and device number (for 1 point). | 20 points | ○ | ○ |
| | | Word | | Sets/resets any specified number of blocks of sixteen bit devices (such as Y and M) and device number (for 16 points). | 10 words (160 points) | | |
| | | | | Writes data to any specified word device (such as Y and M) and device number (for 1 point). | 10 points | | |
| | Monitor data entry | Bit | SADM0 | Defines the bit device (such as Y and M) to be monitored (for 1 point). | 40 points* | ○ | ○ |
| | | Word | | Defines the bit devices (such as Y and M) to be monitored (for 16 points). | 20 words* (320 points) | | |
| | | | | Defines the word device (such as D and R) to be monitored (for 1 point). | 20 points | | |
| | Monitor | Bit | SADM1 | Monitors the device specified in monitor data entry. | | ○ | ○ |
| | | Word | | | | | |

Table 7.2 System Subroutines and Functions

Key, ○ : indicates available.

*: When the A1(E), A2(E), A3(E), A1N, A2N or A3NCPU is used, half the number of points shown above is processed for device X.

When the A0J2 or A3HCPU is used, there is no restriction like above.

- (2) Valid device ranges for device memory transactions are given below:

The device range depends on the PC CPU module used. See the corresponding CPU User's Manual.

| Bit Devices | | | Word Devices | | |
|---------------------|---------------------|---|---------------------------|---------------------|---|
| Device | Device number range | Reresented in: Decimal/ Hexadecimal | Device | Device number range | Reresented in: Decimal/ Hexadecimal |
| Input X | X0000 to X07FF | Hexadecimal | Timer (present value) T | TN000 to TN255 | Decimal |
| Output Y | Y0000 to Y07FF | Hexadecimal | Counter (present value) C | CN000 to CN255 | Decimal |
| Internal relay M | M0000 to M2047 | Decimal | Data register D | D0000 to D1023 | Decimal |
| Latch relay L | L0000 to L2047 | Decimal | Link register W | W0000 to W03FF | Hexadecimal |
| Link relay B | B0000 to B03FF | Hexadecimal | File register R | R0000 to R8191 | Decimal |
| Step relay S | S0000 to S2047 | Decimal | Special register D | D9000 to D9255 | Decimal |
| Annunciator F | F0000 to F0255 | | | | |
| Special relay M | M9000 to M9255 | | | | |
| Timer (Contact) T | TS000 to TS255 | | | | |
| Timer (Coil) T | TC000 to TC255 | | | | |
| Counter (Contact) C | CS000 to CS255 | | | | |
| Counter (Coil) C | CC000 to CC255 | | | | |

Table 7.3 Valid Device Ranges

POINT

- (1) Bit devices and word devices are differentiated as follows.

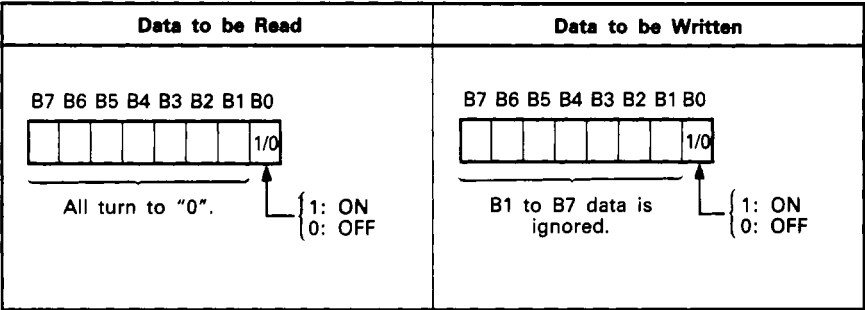
Bit device X, Y, M, L, S, B, F, T (contact), T (coil), C (contact), C (coil)

Word device T (present value), C (present value), D, W, R

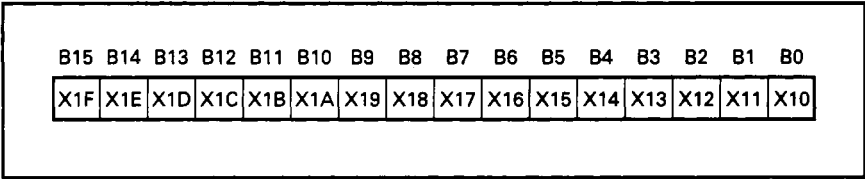
- (2) When reading 16 consecutive bit devices as word data, the bit device number must be a multiple of 16.
- (3) The "0" in the most significant digits of the device number may be specified as spaces (20.).
- (4) M and L ranges are specified in the PC CPU parameters however AD51 device memory transactions regard the two as the same.
- (5) Different device ranges apply when using the AD51 with the A0J2CPU. (See A0J2 Programming Manual)

(3) Bit/word specification

- (a) When a system subroutine has been called with "bit" processing specified, one byte of AD51 memory is used per bit. This is illustrated below, the least significant bit indicates the state of the specified bit device for both reading and writing.



- (b) When a system subroutine has been called with "word" specified, and bit devices are to be processed (i.e. in batches of 16 devices) then one word of AD51 memory is used per 16 bits. This is illustrated below. The head device number is X10 and state of this device is indicated by the least significant bit. The states of the next 15 bits are stored in consecutive bits from Bit B1 to Bit B15.



7.3.2 BASIC program examples

This section gives some BASIC program examples which use system subroutines SADR (batch read), SADW (batch write), SADM0 (monitor data entry), SADM1 (monitor), and SADT (test). The program examples use channel 1.

For details of the system subroutines, refer to the GPC-BASIC Handbooks.

(1) Batch read from device memory (SADR)

[Program example 1]

Program to read data from 16 points, X100 to X10F, to AD51 addresses E000_H to E00F_H as individual bits.

```

100 A = $ F000 ..... Head address for system subroutine IN-
                        PUT data.
110 A:0) = $ FF ..... Defines PC station number as host.
120 A:1) = "B" ..... Specifies bit read.
130 B = $ F002 ..... Sets indirect variable head address.
140 B $ = "X0100" ..... Sets head device to be read using char-
                        acter string variable.
150 C = $ F007 ..... Sets indirect variable head address.
160 C(0) = 16 ..... Sets the number of points to be read as
                        16.
170 C(1) = $ E000 ..... Sets destination head address for data.
180 C(2) = 60 ..... Sets time check period to 60 (600msec).
190 Z = CALL(0, $ 807B, 1, A) ..... Calls system subroutine SADR.
200 IF Z # 0 PRINT "ERROR",
    Z: GOTO 190 ..... Checks for errors in SADR execution.
210 END

```

POINT

(1) When specifying the head device using a character string variable, define the number of points to be read after setting the head device.

(Reason: The character string variable overwrites the bits at the end of the data with "0". This would delete the "number of points" data if this was written first.) For full information, see Section 2.4.4 in the GPC-BASIC Handbook.

(2) The time check period in line 180 should be set in accordance with the "number of scans required for processing" given in Section 3.8 after taking into account any delays which may occur due to other devices accessing the PC CPU.

[Program example 2]

Program to read data from 16 points, X100 to X10F to AD51 addresses E000_H to E001_H in word units. (i.e. batches of 16 bits)

```
100 A= $ F000 ..... Head address for system subroutine IN-
                        PUT data
110 A:0)= $ FF ..... Defines PC station number as host.
120 A:1)= "W" ..... Specifies word read.
130 B= $ F002 ..... Sets indirect variable head address.
140 B $= "X0100" ..... Sets head device to be read using char-
                        acter string variable.
150 C= $ F007 ..... Sets indirect variable head address.
160 C(0)=1 ..... Sets the number of points to be read as
                        1.
170 C(1)= $ E000 ..... Sets destination head address for data.
180 C(2)=60 ..... Sets time check period to 60 (600msec).
190 Z=CALL(0, $ 807B, 1, A) ..... Calls system subroutine SADR.
200 IF Z#0 PRINT "ERROR",
    Z; GOTO 190 ..... Checks for errors in SADR execution.
210 END
```

REMARKS

When "bit read" is specified, one byte of AD51 memory is required per bit of data.
When "word read" is specified, one byte of AD51 memory contains 8 bits of data.
See below:

| Bit Units (Example 1) | Word Units (Example 2) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|--|------------------------|------|------|------|------|------|------|------|------|--|-------|--|--|--|--|--|--|--|-----|------|-------|--|--|--|--|--|--|--|-----|------|-------|--|--|--|--|--|--|--|-----|------|-------|--|--|--|--|--|--|--|--|--|-------|--|--|--|--|--|--|--|-----|------|-------|--|--|--|--|--|--|--|-----|------|-------|--|--|--|--|--|--|--|-----|------|--|--|----|----|----|----|----|----|----|----|--|-------|------|------|------|------|------|------|------|------|--|-------|------|------|------|------|------|------|------|------|--|
| <table><tr><td></td><td>B7</td><td>B6</td><td>B5</td><td>B4</td><td>B3</td><td>B2</td><td>B1</td><td>B0</td><td></td></tr><tr><td>E000H</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>1/0</td><td>X100</td></tr><tr><td>E001H</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>1/0</td><td>X101</td></tr><tr><td>E002H</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>1/0</td><td>X102</td></tr><tr><td>E003H</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>E00DH</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>1/0</td><td>X10D</td></tr><tr><td>E00EH</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>1/0</td><td>X10E</td></tr><tr><td>E00FH</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>1/0</td><td>X10F</td></tr></table> <p>All turn to 0.</p> <p>1: ON 0: OFF</p> | | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 | | E000H | | | | | | | | 1/0 | X100 | E001H | | | | | | | | 1/0 | X101 | E002H | | | | | | | | 1/0 | X102 | E003H | | | | | | | | | | E00DH | | | | | | | | 1/0 | X10D | E00EH | | | | | | | | 1/0 | X10E | E00FH | | | | | | | | 1/0 | X10F | <table><tr><td></td><td>B7</td><td>B6</td><td>B5</td><td>B4</td><td>B3</td><td>B2</td><td>B1</td><td>B0</td><td></td></tr><tr><td>E000H</td><td>X107</td><td>X106</td><td>X105</td><td>X104</td><td>X103</td><td>X102</td><td>X101</td><td>X100</td><td></td></tr><tr><td>E001H</td><td>X10F</td><td>X10E</td><td>X10D</td><td>X10C</td><td>X10B</td><td>X10A</td><td>X109</td><td>X108</td><td></td></tr></table> | | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 | | E000H | X107 | X106 | X105 | X104 | X103 | X102 | X101 | X100 | | E001H | X10F | X10E | X10D | X10C | X10B | X10A | X109 | X108 | |
| | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| E000H | | | | | | | | 1/0 | X100 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| E001H | | | | | | | | 1/0 | X101 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| E002H | | | | | | | | 1/0 | X102 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| E003H | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| E00DH | | | | | | | | 1/0 | X10D | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| E00EH | | | | | | | | 1/0 | X10E | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| E00FH | | | | | | | | 1/0 | X10F | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| E000H | X107 | X106 | X105 | X104 | X103 | X102 | X101 | X100 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| E001H | X10F | X10E | X10D | X10C | X10B | X10A | X109 | X108 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

(2) Batch write to device memory (SADW)

[Program example 1]

Program to write on/off data to Y80 to Y9F from the AD51 as individual bits.

Switch on Y80 to Y8F
Switch off Y90 to Y9F

```

100 A= $E800 ..... Head address for system subroutine IN-
                        PUT data
110 A:0)= $FF ..... Defines PC station number as host.
120 A:1)="B" ..... Specifies bit write.
130 B= $E802 ..... Sets indirect variable head address.
140 B$="Y0080" ..... Sets destination head device using char-
                        acter string variable.
150 C= $E807 ..... Sets indirect variable head address.
160 C(0)=32 ..... Sets the number of points to be written
                        to 32.
170 C(1)= $E000 ..... Sets source data head address.
180 C(2)=30 ..... Sets time check period to 30 (300msec).
190 D= $E000
200 FOR I=0 TO 15
210 D:I)=1
220 D:I+16)=0
230 NEXT I
                        } ..... Generates data to be written to addres-
                        ses E000H to E01FH.
240 Z+CALL(0, $807E, 1, A) ..... Calls system subroutine SADW.
250 IF Z#0 PRINT "ERROR",
      Z; GOTO 240 ..... Check for errors in SADW execution.
260 END

```

[Program example 2]

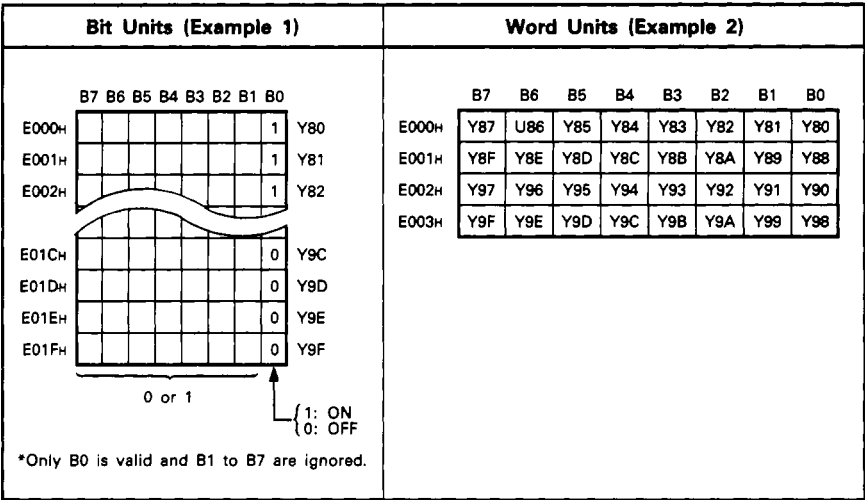
Program to write on/off data to Y80 to Y9F from the AD51 as individual bits.

Switch on Y80 to Y8F
Switch off Y90 to Y9F

```
100 A= $ E800..... Head address for system subroutine IN-
                        PUT data
110 A:0)= $ FF..... Defines PC station number as host.
120 A:1)="W"..... Specifies word write.
130 B= $ E802..... Sets indirect variable head address.
140 B $ ="Y0080"..... Sets destination head device using char-
                        acter string variable.
150 C= $ E807..... Sets indirect variable head address.
160 C(0)=2..... Sets the number of words to be written
                        to 2.
170 C(1)= $ E000..... Sets source data head address.
180 C(2)=30..... Sets time check period to 30 (300msec).
190 @( $ E000)= $ FFFF } Generates data to be written to addres-
200 @( $ E002)=0        } ses E000H to E003H.
210 Z=CALL(0, $ 807E, 1, A)..... Calls system subroutine SADW.
220 IF Z#0 PRINT "ERROR",
    Z; GOTO 210..... Checks for errors in SADW execution.
230 END
```

REMARKS

When "bit write" is specified, one byte of AD51 memory is required per bit of data.
When "word write" is specified, one byte of AD51 memory contains 8 bits of data.
See below:



(3) Data write to any PC device data memory (SADT)

[Program example 1]

Program to write on/off data to the following random bit devices:

Switch on Y115, switch off M340.
Switch off B24C, switch on L875.

```

100 A= $ E000 ..... Head address for system subroutine IN-
                        PUT data
110 FOR I=0 TO 3
120 A(I)= $ E100+I*6 } ..... Sets character string variable head
130 NEXT I             address for destination device.
140 C= $ E100 ..... Sets indirect variable head address.
150 A$(0)="Y0115" ..... Sets device name "Y0115".
160 C:5)=1 ..... Source data for Y115.
170 A$(1)="M0340" ..... Sets device name "M0340".
180 C:11)=0 ..... Source data for M340.
190 A$(2)="B024C" ..... Sets device name "B024C".
200 C:17)=0 ..... Source data for B24C.
210 A$(3)="L0875" ..... Sets device name "L0875".
220 C:23)=1 ..... Source data for L875.
230 B= $ E200 ..... Sets indirect variable head address.
240 B:0)= $ FF ..... Defines PC station number as host.
250 B:1)="B" ..... Specifies "bit write".
260 B:1)=4 ..... Specifies 4 pieces of data to be written.
270 B(2)= $ E100 ..... Sets source data head address.
280 B(3)=70 ..... Sets time check period to 70 (700msec).
290 Z=CALL(0, $ 8081, 1, B) ..... Calls system subroutine SADT.
300 IF Z#0 PRINT "ERROR",
      Z; GOTO 290 ..... Checks for errors in SADT execution.
310 END

```

[Program example 2]

Program to write numerical data to the following random word devices.

D567 ← 0 , R882 ← 1234
W187 ← 751, C49 ← 0

```

100 A= $E000..... Sets indirect variable head address.
110 FOR I=0 TO 3
120 A(I)= $E100+I*7 } ..... Character string variable head address
130 NEXT I              for destination device.
140 C= $E100..... Sets indirect variable head address.
150 A$(0)="D0567"..... Sets device name "D0567".
160 @(C+5)=0..... Sets source data for D567.
170 A$(1)="R0882"..... Sets device name "R0882".
180 @(C+12)=1234..... Sets source data for R882.
190 A$(2)="W0187"..... Sets device name "W0187".
200 @(C+19)=751..... Sets source data for W187.
210 A$(3)="CN049"..... Sets device name "CN049".
220 @(C+26)=0..... Sets source data for CN49.
230 B= $E200..... Sets indirect variable head address.
240 B:0)= $FF..... Defines PC station number as host.
250 B:1)="W"..... Specifies "word write".
260 B:1)=4..... Specifies 4 pieces of data to be written.
270 B(2)= $E100..... Sets source data head address.
280 B(3)=70..... Sets time check period to 70 (700msec).
290 Z=CALL(0, $8081,1,B)..... Calls system subroutine SADT.
300 IF Z#0 PRINT "ERROR",
      Z; GOTO 290..... Checks for errors in SADT execution.
310 END

```

REMARKS

The data is stored in the AD51 in the following formats depending on whether "bit" or "word" has been specified.

| Bit (Example 1) | | | | Word (Example 2) | | | | | | | | |
|-----------------|-----|---|---|------------------|-------|-------|-------|--------------------|-------------|--------------------|--------------------|-----|
| E100H | "Y" | | | Device name | E100H | "D" | | | Device name | | | |
| E101H | "0" | | | | E101H | "0" | | | | | | |
| E102H | "1" | | | | E102H | "5" | | | | | | |
| E103H | "1" | | | | E103H | "6" | | | | | | |
| E104H | "5" | | | | E104H | "7" | | | | | | |
| E105H | 0 | 0 | 0 | 0 | 0 | 0 | 1 | Data to be written | E105H | 0 | Data to be written | |
| E106H | "M" | | | E106H | 0 | | | | | | | |
| E107H | "0" | | | Device name | E107H | "R" | | | Device name | | | |
| E108H | "3" | | | | E108H | "0" | | | | | | |
| E109H | "4" | | | | E109H | "8" | | | | | | |
| E10AH | "0" | | | | E10AH | "8" | | | | | | |
| E10BH | 0 | 0 | 0 | | 0 | 0 | 0 | 0 | | Data to be written | E10BH | "2" |
| ~~~~~ | | | | E10CH | 1234 | | | | | | | |
| E112H | "L" | | | Device name | E10DH | ~~~~~ | | | Device name | | | |
| E113H | "0" | | | | ~~~~~ | | | | | | | |
| E114H | "8" | | | | E115H | "C" | | | | | | |
| E115H | "7" | | | | E116H | "N" | | | | | | |
| E116H | "5" | | | | E117H | "0" | | | | | | |
| E117H | 0 | 0 | 0 | 0 | 0 | 0 | 1 | Data to be written | E118H | "4" | Data to be written | |
| ~~~~~ | | | | E119H | "9" | | | | | | | |
| ~~~~~ | | | | E11AH | 0 | | | Data to be written | E11BH | ~~~~~ | | |
| ~~~~~ | | | | ~~~~~ | | | ~~~~~ | | | | | |

Sets data to be written to
(1 : ON, 0 : OFF).

- (4) Defining device numbers to be monitored. (Before a given device can be monitored using the SADM1 subroutine, it must be specified and "entered" using the following procedure.)

[Program example 1]

Program to specify the following bit devices for monitoring (i.e. monitor data entry)

X7D , Y201 , M178
B3A0, T46 contact, C85 coil

```

100 A= $ F000 ..... Sets indirect variable head address.
110 A$="X007DY0201M0178B03A0TS046CC085" .....
..... Stores device numbers into work area.
120 B= $ E800 ..... Sets indirect variable head address.
130 B:0)= $ FF ..... Defines PC station number as host.
140 B:1)="B" ..... Specifies "bit entry".
150 C= $ E802 ..... Sets indirect variable head address.
160 C(0)=6 ..... Sets the number of points to be entered
to 6.
170 C(1)=A ..... Sets "entry" data head address.
180 C(2)=50 ..... Sets time check period to 50 (500msec).
190 Z=CALL(0, $ 8084, 1, B) ..... Calls system subroutine SADM0.
200 IF Z#0 PRINT "ERROR",
Z; GOTO 190 ..... Checks for errors in SADM0 execution.
210 END

```

[Program example 2]

Program to specify the following word devices for monitoring (or bit devices entered as 16 consecutive device numbers).

X60 to X6F, T80 to T95 contacts, C183 present value
D260 , W175 , R700

```

100 A= $ F000 ..... Sets indirect variable head address.
110 A$="X0060TS080CN183D0260W0175R0700" .....
..... Stores device numbers into work area.
120 B= $ E800 ..... Sets indirect variable head address.
130 B:0)= $ FF ..... Defines PC station number as host.
140 B:1)="W" ..... Specifies "word entry".
150 C= $ E802 ..... Sets indirect variable head address.
160 C(0)=6 ..... Sets the number of points to be entered
to 6.
170 C(1)=A ..... Sets "entry" data head address.
180 C(2)=70 ..... Sets time check period to 70 (700msec).
190 Z=CALL(0, $ 8084, 1, B) ..... Calls system subroutine SADM0.
200 IF Z#0 PRINT "ERROR",
Z; GOTO 190 ..... Checks for errors in SADM0 execution.
210 END

```

POINT

The SADM0 system subroutine (Monitor data entry) is used for both bit and word device and, once entered, is valid for all tasks. These system subroutine enter the devices specified into the OS area where they remain valid until new ones are entered.

- (5) Monitoring of devices (SADM1) specified by the monitor data entry (SADM0)

[Program example 1]

Program to specify and monitor the following bit devices:

X7D , Y201 , M178
B3A0, T46 contact, C85 coil

```

100 A= $ F000 ..... Sets indirect variable head address.
110 A $ = "X007DY0201M0178B03A0TS046CC085" .....
..... Stores device numbers into work area.
120 B= $ E800 ..... Sets indirect variable head address.
130 B:0)= $ FF ..... Defines PC station number as host.
140 B:1)= "B" ..... Specifies "bit entry".
150 C= $ E802 ..... Sets indirect variable head address.
160 C(0)=6 ..... Sets the number of points to be entered
to 6.
170 C(1)=A ..... Sets "entry data" head address.
180 C(2)=50 ..... Sets time check period to 50 (500msec).
190 Z=CALL(0, $ 8084, 1, B) ..... Calls system subroutine SADM0.
200 IF Z#0 PRINT "ERROR",
      Z; GOTO 190 ..... Checks for errors in SADM0 execution.
210 D= $ E900 ..... Sets indirect variable head address.
220 D:0)= $ FF ..... Defines PC station number as host.
230 D:1)= "B" ..... Specifies "bit monitor".
230 D(1)= $ EA00 ..... Data destination head address.
240 D(1)= $ EA00 ..... Data destination head address.
250 D(2)=50 ..... Sets time check period to 50 (500msec).
260 Z=CALL(0, $ 8087, 1, D) ..... Calls system subroutine SADM1.
270 IF Z#0 PRINT "ERROR",
      Z; GOTO 260 ..... Checks for errors in SADM1 execution.
280 END

```

[Program example 2]

Program to specify and monitor the following bit devices (and bit devices entered as 16 consecutive device numbers).

X60 to X6F, T80 to T95 contacts, C183 present value
D260 , W175 , R700

```
100 A= $ F000..... Sets indirect variable head address.
110 A $ ="X0060TS080CN183D0260W0175R0700" .....
..... Stores device numbers into work area.
120 B= $ E800..... Sets indirect variable head address.
130 B:0)= $ FF..... Defines PC station number as host.
140 B:1)="W"..... Specifies "word entry".
150 C= $ E802..... Sets indirect variable head address.
160 C(0)=6..... Sets the number of points to be entered
to 6.
170 C(1)=A..... Sets "entry" data head address.
180 C(2)=50..... Sets time check period to 50 (500msec).
190 Z=CALL(0, $ 8084, 1, B)..... Calls system subroutine SADM0.
200 IF Z#0 PRINT "ERROR",
Z; GOTO 190..... Checks for errors in SADM0 execution.
210 D= $ E900..... Sets indirect variable head address.
220 D:0)= $ FF..... Defines PC station number as host.
230 D:1)= $ 57..... Specifies "word monitor".
240 D(1)= $ EA00..... Data destination head address.
250 D(2)=50..... Sets time check period to 50 (500msec).
260 Z=CALL(0, $ 8087, 1, D)..... Calls system subroutine SADM1.
270 IF Z#0 PRINT "ERROR",
Z; GOTO 260..... Checks for errors in SADM1 execution.
280 END
```

REMARKS

The data is stored in the AD51 in the following formats depending on whether "bit" or "word" has been specified.

| Blt (Example 1) | | | | | | | | | | Word (Example 2) | | | | | | | | | | |
|-------------------------|--|--|--|--|--|--|--|--|-----|-------------------------|-------|-----|-----|-----|-----|-----|-----|-----|-----|-----------------------|
| B7 B6 B5 B4 B3 B2 B1 B0 | | | | | | | | | | B7 B6 B5 B4 B3 B2 B1 B0 | | | | | | | | | | |
| EA00H | | | | | | | | | 1/0 | X7D | EA00H | X67 | X66 | X65 | X64 | X63 | X62 | X61 | X60 | } X60 to X6F |
| EA01H | | | | | | | | | 1/0 | Y201 | EA01H | X6F | X6E | X6D | X6C | X6B | X6A | X69 | X68 | |
| EA02H | | | | | | | | | 1/0 | M178 | EA02H | T87 | T86 | T85 | T84 | T83 | T82 | T81 | T80 | } T80 to T95 contacts |
| EA03H | | | | | | | | | 1/0 | B3A0 | EA03H | T95 | T94 | T93 | T92 | T91 | T90 | T89 | T88 | |
| EA04H | | | | | | | | | 1/0 | T46 contact | EA04H | | | | | | | | | } C183 present value |
| EA05H | | | | | | | | | 1/0 | C85 coil | EA05H | | | | | | | | | |
| | | | | | | | | | | | EA06H | | | | | | | | | } D260 |
| | | | | | | | | | | | EA07H | | | | | | | | | |
| | | | | | | | | | | | EA08H | | | | | | | | | } W175 |
| | | | | | | | | | | | EA09H | | | | | | | | | |
| | | | | | | | | | | | EA0AH | | | | | | | | | } R700 |
| | | | | | | | | | | | EA0BH | | | | | | | | | |

7.4 Extension File Register Read/Write

The following procedures are used to access the extension file registers from the AD51.

7.4.1 System subroutines and functions

The following system subroutines are used by the AD51 to access the PC CPU extension file registers.

(1) System subroutine types and functions

| Item | System Subroutine | Processing | Number of Points Processed per PC CPU, AD51 Transaction | PC CPU State | |
|----------------------------|-------------------|---|---|-----------------------|-----------------------|
| | | | | During STOP | During RUN |
| Batch read | SAER | Reads data from extension file register (for 1 point). | 64 | <input type="radio"/> | <input type="radio"/> |
| Batch write | SAEW | Writes data to extension file register (for 1 point). | | <input type="radio"/> | <input type="radio"/> |
| Test (during random write) | SAET | Writes data to any specified extension file register (for 1 point). | 10 | <input type="radio"/> | <input type="radio"/> |
| Monitor data entry | SAEM0 | Defines the extension file register to be monitored. | 20 | <input type="radio"/> | <input type="radio"/> |
| Monitor | SAEM1 | Monitors the device specified in monitor data entry. | | <input type="radio"/> | <input type="radio"/> |

Key ○ : Indicates available.

Table 7.4 System Subroutines and Functions

(2) Extension file registers

The empty area of the memory cassette has been defined for use as extension file registers in groups of 8K points (16K bytes).

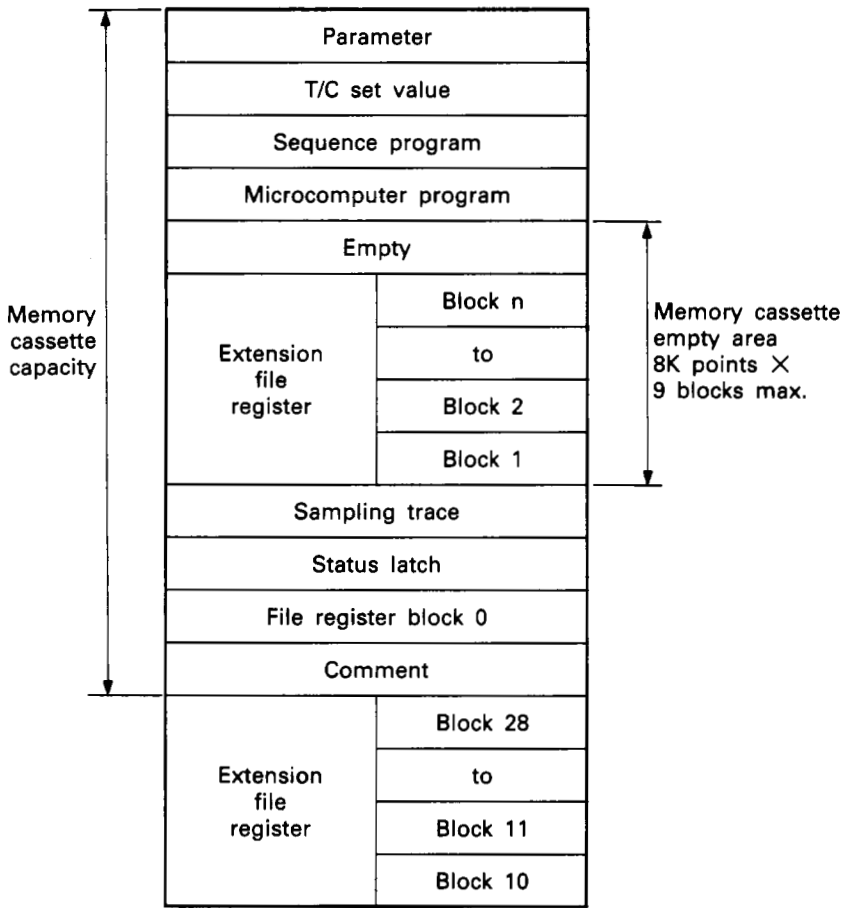
To use this area as extension file registers, the SW-□GHP-UTLPC-FN1 utility program must be stored to the micro-computer program area of the memory cassette.

The utility program automatically defines the empty area in blocks of 8K points as extension file registers.

For full information on the utility program, see the SW-□GHP-UTLP-FN1 Operating Manual.

(3) Extension file register block numbers

When the empty area is defined as extension file registers, block numbers are automatically allocated in groups of 8K points in accordance with the used CPU, memory cassette, program capacity, etc. as shown below:



For more information, see Section 4.8.2 in the SW-60GHP-UTLP-FN1 Operating Manual.

(4) Valid block numbers

The following block numbers may be used in accordance with the memory cassette and CPU used.

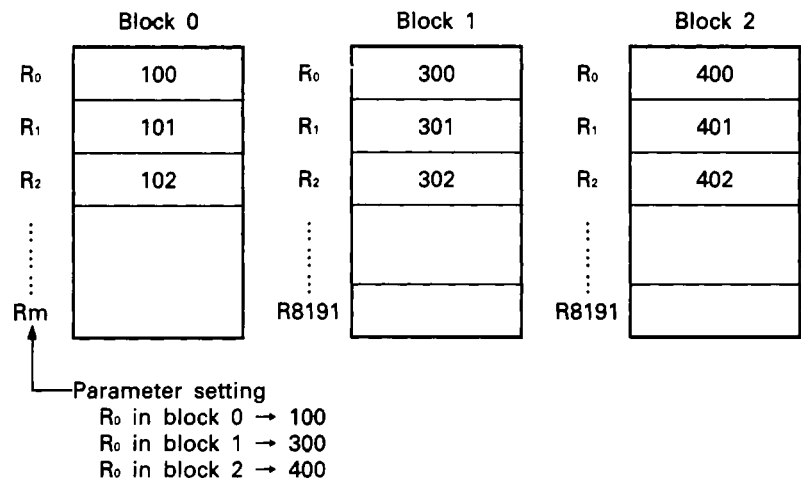
| Memory Cassette Type | | Memory Cassette Capacity | Valid Block Numbers (RAM/ROM operation) | | | Invalid Block Numbers |
|----------------------|----------|--------------------------|---|----------------------------------|--------------------------------|--|
| | | | A2N, A3NCPU | A3HCPU | A2(E), A3(E)CPU | |
| A3NMCA-0 | A3MCA-0 | 16K bytes | Invalid | Invalid | Invalid | |
| A3NMCA-2 | A3MCA-2 | 16K bytes | Invalid | Invalid | Invalid | |
| A3NMCA-4 | A3MCA-4 | 32K bytes | Block 1 only | Block 1 only | Block 1 only | |
| A3NMCA-8 | A3MCA-8 | 64K bytes | Up to block 3 | Up to block 3 | Up to block 3 | |
| — | A3MCA-12 | 96K bytes | Up to block 5 | Up to block 5 | Up to block 5 | Blocks 10, 11 |
| A3NMCA-16 | — | 96K bytes | Up to block 5 Blocks 10, 11 | Up to block 5 Blocks 10, 11 | Up to block 5 Blocks 10, 11 | |
| — | A3MCA-18 | 144K bytes | Up to block 9 | Up to block 9 | Up to block 9 | Blocks 10 to 28 |
| A3NMCA-24 | — | 144K bytes | Up to block 9 Blocks 10 to 12 | Up to block 9 Blocks 10 to 12 | Up to block 9 | A2(E), A3(E)CPU : Blocks 10 to 28 A2N, A3NCPU : Blocks 13 to 28 A3HCPU |
| A3NMCA-40 | — | 144K bytes | Up to block 9 Blocks 10 to 20 | Up to block 9 Blocks 10 to 20 | Up to block 9 | A2(E), A3(E)CPU : Blocks 10 to 28 A2N, A3NCPU : Blocks 21 to 28 A3HCPU |
| A3NMCA-56 | — | 144K bytes | Up to block 9 Blocks 10 to 20 | Up to block 9 Blocks 10 to 20 | Up to block 9 | A2(E), A3(E)CPU : Blocks 10 to 28 A2N, A3NCPU : Blocks 21 to 28 |

Table 7.5 Valid Block Numbers

- (a) As the A3NMCA-12 is processed as the A3NMCA-16, and the A3NMCA-18, A3NMCA-24 and A3NMCA-40 processed as the A3NMCA-56, specifying an invalid block number does not lead to an error but the execution result of the system subroutine is invalid.
- (b) Block 9 may be accessed from the AD51 but cannot be accessed by the sequence program.
- (c) The AD51 is allowed to access all valid register blocks but the sequence program is only allowed to access the number of file register points set in the parameters.

(5) Relation between file registers and block numbers

To access the file register in any block, specify the required block number and file register device number, then execute the required system subroutine.



(6) Relation between device numbers and set values

| Device Number | Set Value |
|---------------|-----------|
| R0 | R0000 |
| R1 | R0001 |
| to | to |
| R8190 | R8190 |
| R8191 | R8191 |

7.4.2 BASIC program examples

Some BASIC program examples are given below which use system subroutines SAER (batch read), SAEW (batch write), SAET (test), SAEM0 (monitor data entry) and SAEM1 (monitor). The program examples use channel 1.

Full information on the system subroutines is given in the GPC-BASIC Handbook.

(1) Batch read from extension file registers (SAER)

Example: Program to read data from 16 points, file registers R100 to R115 in block 2, to addresses E000_H to E01F_H.

```

100 A = $ F000 ..... Head address for system subroutine IN-
                        PUT data.
110 A:0) = $ FF ..... Defines PC station number as host.
120 A:1) = 2 ..... Specifies block 2.
130 B = $ F002 ..... Sets indirect variable head address.
140 B $ = "R0100" ..... Sets head device to be read using char-
                        acter string variable.
150 C = $ F007 ..... Sets indirect variable head address.
160 C(0) = 16 ..... Sets the number of points to be read as
                        16.
170 C(1) = $ E000 ..... Sets destination head address for data.
180 C(2) = 60 ..... Sets time check period to 60 (600ms).
190 Z = CALL(0, $ 80BD, 1, A) ..... Calls system subroutine SAER.
200 IF Z # 0 PRINT "ERROR",
    Z; GOTO 190 ..... Checks for errors in SAER execution.
210 END

```

POINT

- (1) When specifying the head device using a character string variable, define the number of points to be read after setting the head device. (Reason: The character string variable overwrites the bits at the end of the data with "0". This would delete the "number of points" data if this was written first.)
- (2) The time check period in line 180 should be set in accordance with the "number of scans required for processing" given in Section 3.8 after taking into account any delays which may occur due to other devices accessing the PC CPU.

(2) Batch write to extension file registers (SAEW)

Example: Program to write data from addresses E000_H-E01F_H to file registers R200-R215 in block 5 (16 points).

```

100 A= $ F000 ..... Head address for system subroutine IN-
                        PUT data.
110 A:0)= $ FF ..... Defines PC station number as host.
120 A:1)=5 ..... Specifies block 5.
130 B= $ F002 ..... Sets indirect variable head address.
140 B$="R0200" ..... Sets destination head device using char-
                        acter string variable.
150 C= $ F007 ..... Sets indirect variable head address.
160 C(0)=16 ..... Sets the number of points to be written
                        as 16.
170 C(1)= $ E000 ..... Sets source head address for data.
180 C(2)=60 ..... Sets time check period to 60 (600ms).
190 Z=CALL(0, $ 80C0, 1, A) ..... Calls system subroutine SAEW.
200 IF Z#0 PRINT "ERROR",
    Z; GOTO 190 ..... Checks for errors in SAEW execution.
210 END

```

(3) Data write to any specified extension file registers (SAET)

Example: Program to write 246 to file register R100 in block 5 and 1234 to R8191 in block 7.

```

100 A= $ FE00 ..... Sets indirect variable head address.
110 A:0)=5 ..... Specifies block 5.
120 A:8)=7 ..... Specifies block 7.
130 B= $ FE01 ..... Sets indirect variable head address.
140 B$="R0100" ..... Sets file register R100.
150 C= $ FE09 ..... Sets indirect variable head address.
160 C$="R8191" ..... Sets file register R8191.
170 D= $ FE06 ..... Sets indirect variable head address.
180 D(0)=246 ..... Sets data written to R100.
190 D(4)=1234 ..... Sets data written to R8191.
200 E= $ F000 ..... Sets indirect variable head address.
210 E:0)= $ FF ..... Defines PC station number as host.
220 F= $ F001 ..... Sets indirect variable head address.
230 F(0)=2 ..... Sets the number of points to be written
                        as 2.
240 F(1)= $ FE00 ..... Sets destination head address for data.
250 F(2)=60 ..... Sets time check period to 60 (600ms).
260 Z=CALL(0, $ 80C3, 1, E) ..... Calls system subroutine SAET.
270 IF Z#0 PRINT "ERROR",
    Z; GOTO 260 ..... Checks for errors in SAET execution.
280 END

```

- (4) Defining file registers to be monitored. (SAEM0) (Before a given file register can be monitored using the SAEM1 subroutine, it must be specified and entered using the following procedure.)

Example: Program to specify file register R50 in block 1 and R100 in block 7.

```

100 A= $FE00..... Sets indirect variable head address.
110 A:0)=1 ..... Specifies block 1.
120 B= $FE01 ..... Sets indirect variable head address.
130 B$="R0050" ..... Sets file register R50.
140 A:6)=7 ..... Specifies block 7.
150 C= $FE07 ..... Sets indirect variable head address.
160 C$="R0100" ..... Sets file register R100.
170 D= $F000 ..... Sets indirect variable head address.
180 D:0)= $FF ..... Defines PC station number as host.
190 E= $F001 ..... Sets indirect variable head address.
200 E(0)=2 ..... Sets the number of points to be entered
                    as 2.
210 E(1)= $FE00 ..... Sets destination head address for data.
220 E(2)=60 ..... Sets time check period to 60 (600ms).
230 Z=CALL(0, $80C6, 1, D) ..... Calls system subroutine SAEM0.
240 IF Z#0 PRINT "ERROR",
    Z; GOTO 230 ..... Checks for errors in SAEM0 execution.
250 END

```

- (5) Monitoring of file registers (SAEM1) specified by the monitor data entry (SAEM0)

Example: Program to specify and monitor the file registers entered by using the program in (4).

```

100 A= $FE00
    to
230 Z=CALL(0, $80C6, 1, D) } ..... Monitor data entry in accordance with
240 IF Z#0 PRINT "ERROR",   } (4).
    Z; GOTO 230
250 F= $F007 ..... Sets indirect variable head address
260 F:0)= $FF ..... Defines PC station number as host.
270 G= $F008 ..... Sets indirect variable head address.
280 G(0)= $F100 ..... Sets destination head address for data.
290 G(1)=60 ..... Sets time check period to 60 (600ms).
300 Z=CALL(0, $80C9, 1, F) ..... Calls system subroutine SAEM1.
310 IF Z#0 PRINT "ERROR",
    Z; GOTO 300 ..... Checks for errors in SAEM1 execution.
320 END

```

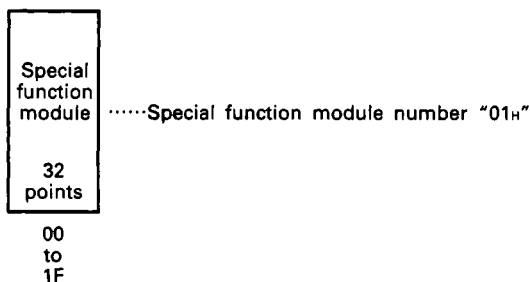

| Special Function Module | Buffer Memory Head Address (Hexadecimal) |
|--|--|
| A68AD analog-to-digital converter module | 80 _H |
| A62DA digital-to-analog converter module | 10 _H |
| A84AD analog/digital converter module | 10 _H |
| AD61(S1) high-speed counter module | 82 _H |
| AD71(S1) positioning module | 200 _H |
| AD72 positioning module | 200 _H |
| AD51 intelligent communication module | 800 _H |
| AJ71C24-S3 computer link module | 1000 _H |
| A81CPU PID control module | 200 _H |

Table 7.8 Special Function Module Head Addresses

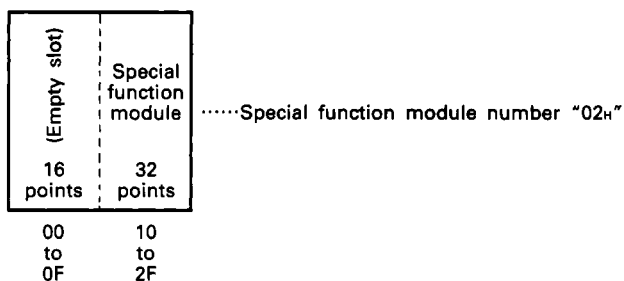
(3) Special function modules and module numbers

The special function module number (hexadecimal) specified by the AD51 is the first two digits of the three-digit final I/O address of the special function module with respect to the PC CPU.

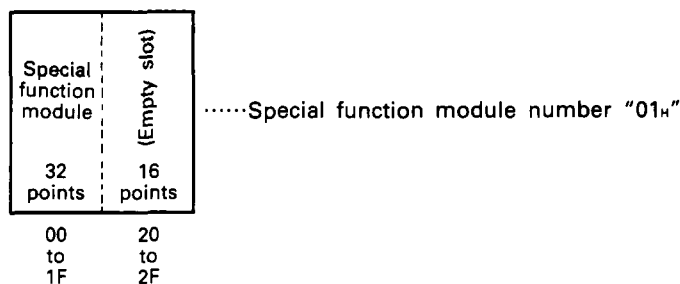
(a) Module occupying one slot (such as AD61, A68AD)



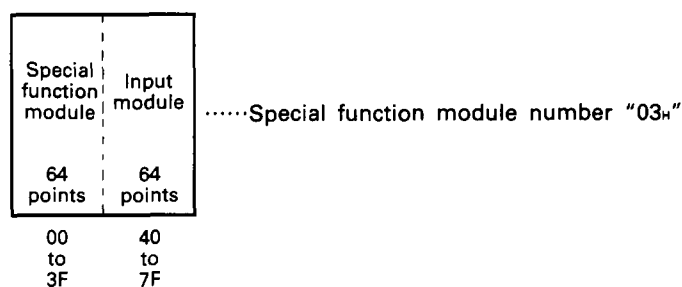
(b) Module occupying two slots and using the first slot as empty (e.g. AD72, A84AD)



- (c) Module occupying two slots and using the second slot as empty



- (d) Module occupying two slots as a special function module and an input module (A81CPU)



| Special Function Module | Module Number of Module on Slot 0 |
|--|---|
| A68AD analog-to-digital converter module | 01 _H |
| A62DA digital-to-analog converter module | 01 _H |
| A84AD analog/digital converter module | 02 _H |
| AD61(S1) high-speed counter module | 01 _H |
| AD71(S1) positioning module | 01 _H |
| AD72 positioning module | 02 _H |
| AD51 intelligent communication module | 02 _H |
| AJ71C24-S3 computer link module | 01 _H |
| A81CPU PID control module | 03 _H |

Table 7.9 Special Function Modules and Module Numbers

The module number of the special function module used on a MELSECNET remote station depends on the link parameters set to the MELSECNET master station.

| L/R NO. | $M \leftarrow L$ | | $M \rightarrow R$ | $M \leftarrow R$ | $M \rightarrow L/R$ | | $M \leftarrow L/R$ | |
|---------|------------------|-------|-------------------|------------------|---------------------|---------|--------------------|---------|
| | B | W | W | W | Y | X/Y | X | Y/X |
| R1 | ----- | ----- | 29C-309 | 0F9-15E | 400-48F | 000-08F | 430-44F | 030-04F |
| R2 | ----- | ----- | 215-24F | 080-0A3 | 510-67F | 010-17F | 500-65F | 000-15F |
| R3 | ----- | ----- | 1B6-214 | 15F-1B5 | 270-32F | 050-10F | 220-28F | 000-06F |
| | - | - | - | - | - | - | - | - |
| | - | - | - | - | - | - | - | - |
| | - | - | - | - | - | - | - | - |
| | - | - | - | - | - | - | - | - |
| | - | - | - | - | - | - | - | - |

Special function module
number "H44"

7.5.2 BASIC program examples

The following BASIC program examples use system subroutines SATR (batch read) and SATW (batch write) to access the special function module buffer memory.

The program examples use channel 1.

For full information on the system subroutines, see the GPC-BASIC Handbook.

(1) Batch read from buffer memory (SATR)

Example: Program to read 4-byte data from buffer memory address 10_H (94_H) of the A68AD at I/O addresses 80_H-9F_H to address E000_H.

```

100 A= $ F000 ..... Head address for system subroutine IN-
                        PUT data.
110 A:0)= $ FF ..... Defines PC station number as host.
120 A:1)= $ 09 ..... Specifies module number.
130 B= $ F002 ..... Sets indirect variable head address.
140 B(0)= $ 94 ..... Sets buffer memory head address.
150 B:2)=0
160 C= $ F005 ..... Sets indirect variable head address.
170 C(0)=4 ..... Sets the number of bytes to be read as 4.
180 C(1)= $ E000 ..... Sets destination head address for data.
190 C(2)=60 ..... Sets time check period to 60 (600ms).
200 Z=CALL(0, $ 80D8, 1, A) ..... Calls system subroutine SATR.
210 IF Z#0 PRINT "ERROR",
    Z; GOTO 200 ..... Checks for errors in SATR execution.
220 END

```

(2) Batch write to buffer memory (SATW)

Example: Program to write 4090 to buffer memory address 0_H (10_H, 11_H) of the A62DA at I/O addresses 0A_H-0B_H.

```

100 A= $ EE00 ..... Sets indirect variable head address.
110 A(0)=2000 ..... Sets data "2000" to be written.
120 B= $ E000 ..... Sets indirect variable head address.
130 B:0)= $ FF ..... Defines PC station number as host.
140 B:1)= $ 0B ..... Specifies module number.
150 C= $ E002 ..... Sets indirect variable head address.
160 C(0)= $ 10 ..... Sets buffer memory head address.
170 C:2)=0
180 D= $ E005 ..... Sets indirect variable head address.
190 D(0)=2 ..... Sets the number of bytes to be written as
                        2.
200 D(1)= $ EE00 ..... Sets source head address for data.
210 D(2)=60 ..... Sets time check period to 60 (600ms).
220 Z=CALL(0, $ 80DB, 1, B) ..... Calls system subroutine SATW.
230 IF Z#0 PRINT "ERROR",
    Z; GOTO 220 ..... Checks for errors in SATW execution.
240 END

```

7.6 Read/Write of Sequence Program and T/C Set Values

This section describes procedures for reading and writing PC CPU sequence programs and T/C set values from the AD51.

7.6.1 System subroutines and functions

The following system subroutines are used to read and write sequence programs and T/C set values.

(1) System subroutines and functions

| Item | | | System Subroutine | Processing | Amount of information processed per PC — AD51 transaction. | PC CPU State | |
|------------------|------------------|------|-------------------|--|--|--------------|-----|
| | | | | | | Stop | Run |
| Sequence program | Read | Main | SAAR | Reads main sequence program. | 64 steps | ○ | ○ |
| | | Sub | | Reads subsequence program. | | | |
| | Write | Main | SAAW | Writes main sequence program. | | ○ | ×* |
| | | Sub | | Writes subsequence program. | | | |
| Parameter | Read | | SAPR | Reads parameters of PC CPU. | 128 bytes | ○ | ○ |
| | Write | | SAPW | Writes parameters of PC CPU. | | ○ | × |
| | Analysis request | | SAPS | Causes PC CPU to recognize and check rewritten parameters. | | ○ | × |

Key : ○ Available
 × Unavailable
 * For T/C only

Table 7.2 System Subroutines and Functions

* A subprogram may be written while a main program is running and vice versa using appropriate control of M9050 and M9051. The A3N and A3HCPUs are not provided with M9050.

POINT

- (1) Sequence programs should be read and written in the range set in the parameters. PC CPU data may be corrupted if programs are written outside the set range.
- (2) An input data error is returned if the specified parameter read/write capacity is outside the allowed range (16 bytes in the A0J2CPU, 3K bytes in the A1(E), A2(E), A3(E), A1N, A2N, A3N, and A3HCPUs).

(2) T/C set values and program addresses

Specify T/C set values and program step numbers as shown in the following table.

| Sequence Program | Address |
|--|---|
| T0 set value T1 set value to T255 set value | FE00 _H FE01 _H to FEFF _H |
| C0 set value C1 set value to C255 set value | FF00 _H FF01 _H to FFFF _H |
| Step 0 Step 1 to Step 30719 (30K) | 0000 _H 0001 _H to 77FF _H |

Calculation of specified address:

$$\text{Timer } T_m = \text{FE00}_H + n$$

$$\text{Counter } C_m = \text{FF00}_H + n$$

where, m = device number

n = hexadecimal value of device number

(3) T/C set value

Read/write data of T/C set values is expressed in hexadecimal as shown in the following table.

| Specification with Constant | Set Value | Specification with D Register | Set Value |
|-----------------------------|-------------------|-------------------------------|-------------------|
| K0 | 0000 _H | D0 | 8000 _H |
| K1 | 0001 _H | D1 | 8002 _H |
| K2 | 0002 _H | D2 | 8004 _H |
| to | | to | to |
| K32766 | 7FFE _H | D1022 | 81FC _H |
| K32767 | 7FFF _H | D1023 | 81FE _H |

Calculation of set value:

$$K_m = 000_H + n$$

$$D_m = 8000_H + 2n$$

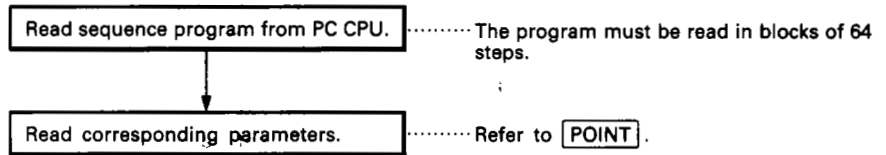
where, m = device number

n = hexadecimal value of device number

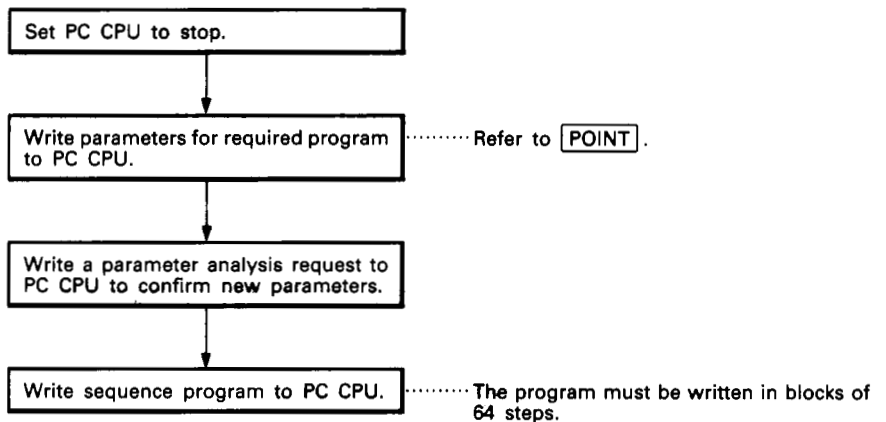
7.6.2 Read and write procedures

Parameter data should not be separated from the sequence program when it is read or written by the AD51. Use the following procedures:

(1) Sequence program read



(2) Write of sequence programs

**POINT**

When reading or writing sequence programs, always, ensure that the parameters match the program. Mismatches of parameters and programs will result in misoperation.

7.6.3 BASIC program example

This section gives the procedure for transferring sequence programs and T/C set values between the PC CPU and the AD51.

(1) Sequence program read (SAAR, SAPR)

[Program example]

Program to read the PC CPU main program to the following AD51 memory areas.

Sequence program (2K steps) → Channel 2 E000_H to EFFF_H
 Parameter (3K bytes) → Channel 2 D000_H to DBFF_H

```

100 Z=ZWR1(2, $ F800, $ FF).....Defines PC station number as host.
110 Z=ZWR1(2, $ F801, $ 4D) .....Sets main, "M", sequence program.
120 Z=ZWR2(2, $ F804, 64).....Sets the number of steps to be read to
                                64.
130 Z=ZWR2(2, $ F808, 60).....Sets time check period to 60 (600m
                                sec).
140 A= $ E000.....Sets data destination head address to
                                variable A.

150 FOR I=0 TO 2047 STEP 64
160 Z=ZWR2(2, $ F802, I) .....Sets head step number of sequence
                                program to be read.
170 Z=ZWR2(2, $ F806, A).....Sets data destination head address.
180 Z=CALL(0, $ 808A, 2, $ F800).....Reads sequence program.
185 IF Z#0 PRINT "ERROR",
    Z; GOTO 180.....Checks for errors in SAAR execution.
190 A=A+128.....Adds 128 (64 steps) to head address
                                of area to be read.

200 NEXT I
210 Z=ZWR1(2, $ F810, $ FF).....Defines PC station number as host.
220 Z=ZWR2(2, $ F814, 128).....Sets read byte length to 128.
230 Z=ZWR2(2, $ F818, 60).....Sets time check period to 60
                                (600msec).

240 FOR I=0 TO $ BFF STEP $ 80
250 Z=ZWR2(2, $ F811, I) }.....Sets parameter read head address.
260 Z=ZWR1(2, $ F813, 0) }
270 Z=ZWR2(2, $ F816, $ D000+I).....Sets data destination head address.
280 Z=CALL(0, $ 8090, 2, $ F810).....Reads parameters.
285 IF Z#0 PRINT "ERROR",
    Z; GOTO 280.....Checks for errors in SAPR execution.
290 NEXT I
300 END
  
```


(2) Sequence program write

[Program example]

Program to write the main sequence program and parameters from the AD51 to the PC CPU.

Channel 2 E000_H to EFFF_H → Sequence program (2K steps)
 Channel 2 D000_H to DBFF_H → Parameters (3K bytes)

```

100 Z=CALL(0, $ 8030, $ FF, 70) .....Checks PC CPU RUN/STOP status.
110 IF Z=1 LOCATE 5, 10;PRINT
    "CPU RUN"; GOTO 100 .....Indicates whether the CPU is running.
120 IF Z#0 GOTO
    [To error processing] .....Detects any error in SKC execution
                                and moves to a suitable part of the
                                program.
130 Z=ZWR1(2, $ F000, $ FF) .....Defines the PC station number as
                                host.
140 Z=ZWR2(2, $ F004, 128) .....Sets byte length to be written to 128.
150 Z=ZWR2(2, $ F008, 70) .....Sets time check period to 70
                                (700msec).
160 FOR I=0 TO $ BFF STEP $ 80
170 Z=ZWR2(2, $ F001, I) } .....Sets parameter write head address.
180 Z=ZWR1(2, $ F003, 0) }
190 Z=ZWR2(2, $ F006, $ D000+I) .....Sets source data head address.
200 Z=CALL(0, $ 8093, 2, $ F000) .....Writes parameters.
210 IF Z#0 PRINT "ERROR",
    Z; GOTO 200 .....Checks for errors in SAPW execution.
220 NEXT I
230 Z=CALL(0, $ 8096, $ FF, 70) .....Parameter analysis request.
240 IF Z#0 PRINT "ERROR",
    Z; GOTO 230 .....Checks for errors in SAPS execution.
250 Z=ZWR1(2, $ F010, $ FF) .....Defines the PC station number as
                                host.
260 Z=ZWR1(2, $ F011, $ 4D) .....Sets main, "M", sequence program.
270 Z=ZWR2(2, $ F014, 64) .....Sets the number of steps to be written
                                to 64.
280 Z=ZWR2(2, $ F018, 70) .....Sets time check period to 70
                                (700msec).
290 A= $ E000 .....Sets data source head address to
                                variable A.
300 FOR I=0 TO 2047 STEP 64
310 Z=ZWR2(2, $ F012, I) .....Sets head step number of sequence
                                program to be written.
320 Z=ZWR2(2, $ F016, A) .....Sets data source head address.
330 Z=CALL(0, $ 808D, 2, $ F010) .....Writes sequence program.
340 IF Z#0 PRINT "ERROR",
    Z; GOTO 330 .....Checks for errors in SAAW execution.
350 A=A+128 .....Adds 128 to sequence program write
                                head address.
360 NEXT I
370 END
  
```

(3) T/C set values read.

[Program example]

Program for reading set values of T0 to T31 and C64 to C79 to memory addresses B000_H to B03F_H and B040_H to B05F_H in channel 2.

```

100 Z=ZWR1(2, $E700, $FF).....Defines PC station number as host.
110 Z=ZWR1(2, $E701, $4D).....Sets main, "M", sequence program.
120 Z=ZWR2(2, $E702, $FE00).....Sets read head step to T0.
130 Z=ZWR2(2, $E704, 32).....Sets the number of points to be read
                                to 32.
140 Z=ZWR2(2, $E706, $B000).....Sets data destination head address.
150 Z=ZWR2(2, $E708, 50).....Sets time check period to 50
                                (500msec).
160 Z=CALL(0, $808A, 2, $E700).....Reads T0 to T31 set values.
170 Z=ZWR2(2, $E702, $FF40).....Sets read head step to C64.
180 Z=ZWR2(2, $E704, 16).....Sets the number of points to be read
                                to 16.
190 Z=ZWR2(2, $E706, $B040).....Sets data destination head address.
200 Z=CALL(0, $808A, 2, $E700).....Reads C64 to C79 set values.
210 IF Z#0 PRINT "ERROR",
    Z; GOTO 200.....Checks for errors in SAAR execution.
220 END

```

7.7 Microcomputer Program Read/Write

PC CPU microcomputer programs are read and written from the AD51 in the following procedures.

7.7.1 System subroutines and functions

The following system subroutines are used to read and write microcomputer programs.

(1) System subroutine types and functions

| Item | | System Subroutine | Processing | Number of Points Processed per PC CPU, AD51 Transaction | PC CPU State | |
|-------|------|-------------------|------------------------------------|---|--------------|------------|
| | | | | | During STOP | During RUN |
| Read | Main | SAMR | Reads main microcomputer program. | 128 bytes | ○ | ○ |
| | Sub | | Reads sub microcomputer program. | | | |
| Write | Main | SAMW | Writes main microcomputer program. | | ○ | × * |
| | Sub | | Writes sub microcomputer program. | | | |

Key, ○ : Available

Table 7.11 System Subroutines and Functions

* When the A3CPU or A3ECPU is used, a subprogram may be written while a main program is running and vice versa using appropriate control of M9050 and M9051.
The A3N and A3HCPUs are not provided with M9050.

POINT

The M9050 circuit must be deleted if any A3CPU or A3ECPU program is utilized for the A3N or A3HCPU.
Reason: If M9050 is on, any program cannot be written during run of the A3N or A3HCPU.

(2) Microcomputer program read/write head address

Specify the microcomputer program read/write head address using the corresponding offset address as indicated below:

| Microcomputer Program | Offset Address |
|-----------------------|---------------------|
| Step 0 | 0000 _H |
| Step 1 | 0001 _H |
| to | to |
| Step 59391 (58K) | E7FF _H * |

*: This address assumes that the microcomputer program capacity has been set to 58K bytes in the parameter.

The (head address) + (number of bytes processed) should be ((microcomputer program capacity set in parameter) - 1 byte) max.

(Head address) + (number of bytes processed) \leq ((microcomputer program capacity set in parameter) - 1 byte)

POINT

When reading or writing microcomputer programs, always ensure that the parameters match the programs. Mismatches of parameters and programs may corrupt the PC CPU data.

7.7.2 BASIC program examples

The following program examples transfer microcomputer programs between the PC CPU and AD51.

(1) Microcomputer program read (SAMR)

Example: Program to read PC CPU main microcomputer program to the following AD51 memory area:

Microcomputer program (4K bytes) → Channel 2 E000_H to EFFF_H

```

100 Z=ZWR1(2, $ F800, $ FF).....Defines PC station number as host.
110 Z=ZWR1(2, $ F801, $ 4D) .....Sets main, "M", microcomputer
                                   program.
120 Z=ZWR2(2, $ F804, 64).....Sets the number of bytes to be read
                                   as 64.
130 Z=ZWR2(2, $ F808, 60).....Sets time check period to 60 (600ms).
140 A= $ E000.....Sets indirect variable head address.
150 FOR I=0 TO 4095 STEP 64
160 Z=ZWR2(2, $ F802, I) .....Sets head address of microcomputer
                                   program to be read.
170 Z=ZWR2(2, $ F806, A).....Sets data destination head address.
180 Z=CALL(0, $ 80CC, 2, $ F800).....Calls system subroutine SAMR.
190 IF Z#0 PRINT "ERROR",
    Z; GOTO 180.....Checks for errors in SAMR execution.
200 A=A+64.....Adds 64 to data destination head
                                   address.

210 NEXT I
220 END

```

(2) Microcomputer program write (SAMW)

Example: Program to write the main microcomputer program from the AD51 to the PC CPU.

Channel 2 E000_H to EFFF_H → Microcomputer program (4K bytes)

```

100 Z=CALL(0, $ 8030, $ FF, 70) .....Checks PC CPU RUN/STOP status.
110 IF Z=1 LOCATE 5, 10;PRINT
    "CPU RUN"; GOTO 100 .....Indicates whether the CPU is running.
120 IF Z#0 GOTO
    [To error processing] .....Detects any error in SKC execution
                                and moves to a suitable part of the
                                program.
130 Z=ZWR1(2, $ F000, $ FF) .....Defines PC station number as host.
140 Z=ZWR1(2, $ F001, $ 4D) .....Sets main, "M", microcomputer
                                program.
150 Z=ZWR2(2, $ F004, 64) .....Sets the number of bytes to be writ-
                                ten as 64.
160 Z=ZWR2(2, $ F008, 60) .....Sets time check period to 60 (600ms).
170 A= $ E000 .....Sets indirect variable head address.
180 FOR I=0 TO 4095 STEP 64
190 Z=ZWR2(2, $ F002, I) .....Sets head address of microcomputer
                                program to be written.
200 Z=ZWR2(2, $ F006, A) .....Sets data source head address.
210 Z=CALL(0, $ 80CF, 2, $ F000) .....Calls system subroutine SAMW.
220 IF Z#0 PRINT "ERROR",
    Z; GOTO 210 .....Checks for errors in SAMW execution.
230 A=A+64 .....Adds 64 to data source head address.
240 NEXT I
250 END

```

7.8 Comment Read/Write

Comments are read and written in the following procedures:

7.8.1 System subroutines and functions

The following system subroutines are used to read and write PC CPU comments from the AD51.

(1) System subroutine types and functions

| Item | System Subroutine | Processing | Number of Points Processed per PC CPU, AD51 Transaction | PC CPU State | |
|-------------|-------------------|------------------|---|-----------------------|-----------------------|
| | | | | During STOP | During RUN |
| Batch read | SACR | Reads comments. | 128 bytes | <input type="radio"/> | <input type="radio"/> |
| Batch write | SACW | Writes comments. | | <input type="radio"/> | <input type="radio"/> |

Key, ○ : Available

Table 7.12 System Subroutines and Functions

(2) Comment read/write head address

Specify the comment read/write head address using the offset address.

The (head address) + (number of bytes processed) should be equal to or less than the comment capacity set in the parameter.

$$(\text{Head address}) + (\text{number of bytes processed}) \leq (\text{comment capacity})$$

POINT

When reading or writing comments, always ensure that the parameters match the comments. Mismatches of parameters and comments may corrupt the PC CPU data.

7.8.2 BASIC program examples

The following program examples transfer comments between the PC CPU and AD51.

(1) Comment read (SACR)

Example: Program to read PC CPU comments to the following AD51 memory area:

192 comment points (4K bytes) → Channel 2 8000_H to 8FFF_H

```
100 Z=ZWR1(2, $ F800, $ FF).....Defines PC station number as host.
110 Z=ZWR2(2, $ F803, 64).....Sets the number of bytes to be read
                                   as 64.
120 Z=ZWR2(2, $ F807, 60).....Sets time check period to 60 (600ms).
130 A= $ 8000.....Sets indirect variable head address.
140 FOR I=0 TO 4095 STEP 64
150 Z=ZWR2(2, $ F801, I).....Sets head address of comments to be
                                   read.
160 Z=ZWR2(2, $ F805, A).....Sets data destination head address.
170 Z=CALL(0, $ 80D2, 2, $ F800).....Calls system subroutine SACR.
180 IF Z#0 PRINT "ERROR",
    Z; GOTO 170.....Checks for errors in SACR execution.
190 A=A+64.....Adds 64 to data destination head
                                   address.

200 NEXT I
210 END
```

(2) Comment write (SACW)

Example: Program to write comments from the AD51 to the PC CPU.

Channel 2 8000_H to 8FFF_H → 192 comment points (4K bytes)

```

100 Z=ZWR1(2, $ F900, $ FF).....Defines PC station number as host.
110 Z=ZWR2(2, $ F903, 64).....Sets the number of bytes to be writ-
                                ten as 64.
120 Z=ZWR2(2, $ F907, 60).....Sets time check period to 60 (600ms).
130 A= $ 8000 .....Sets indirect variable head address.
140 FOR I=0 TO 4095 STEP 64
150 Z=ZWR2(2, $ F901, I) .....Sets head address of comments to be
                                written.
160 Z=ZWR2(2, $ F905, A).....Sets data source head address.
170 Z=CALL(0, $ 80D5, 2, $ F900).....Calls system subroutine SACW.
180 IF Z#0 PRINT "ERROR",
    Z; GOTO 170.....Checks for errors in SACW execution.
190 A=A+64 .....Adds 64 to data source head address.
200 NEXT I
210 END

```


7.9 Interrupt from PC CPU to AD51

The interrupt signal from the PC CPU to the AD51 is valid on its rise (i.e. the AD51 waits for the leading edge of the signal).

[How to use]

To enable the interrupt facility, the relevant task should be set to "start at interrupt from PC CPU" on the multi task setting screen. The AD51 will then run that task when it receives the interrupt signal.

Once it has been started the interrupt task will operate until the **END** instruction is executed. The interrupt is re-enabled after the **END** instruction is executed.

[Program example]

The following P.C. program will call the designated interrupt task when X01 turns on.

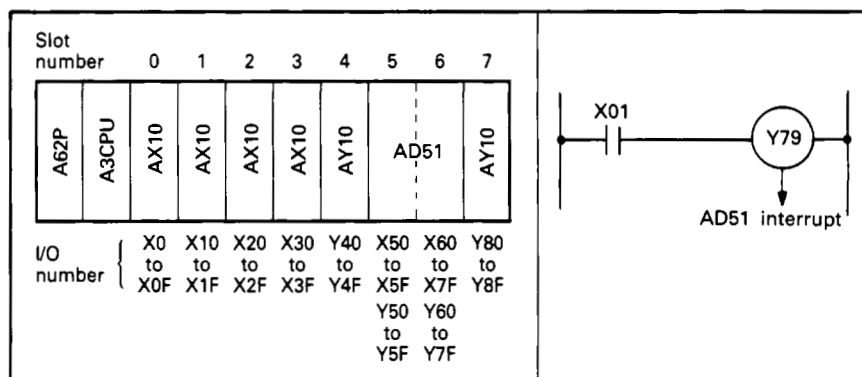


Fig. 7.5 System Configuration and Sequence Program

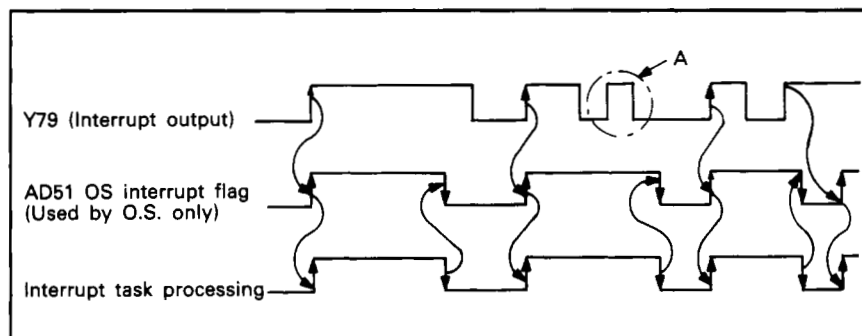


Fig. 7.6 Timing Chart

POINT

- (1) Any interrupt signal given to the AD51 while an interrupt program is running (i.e. interrupt flag is ON) will be ignored (See area A in Fig. 7.6.)
- (2) Only one task may be specified as "interrupt start". Setting any more generally leads to ORST error.

7.10 Interrupt from AD51 to PC CPU

System subroutine "SIT" causes the AD51 to interrupt the PC CPU and allows AD51 interrupt sequence programs to be executed. The A0J2CPU cannot be interrupted by the AD51.

The PC CPU has interrupt pointers, I16 to I23, which are assigned to interrupt signals generated by special function modules in order of I/O allocation.

For details, refer to the CPU Unit User's Manual and Programming Manual.

In the system configuration shown in Fig. 7.7, when AD51 No. 1 interrupts the PC CPU, the interrupt program designated by pointer I16 is executed. When AD51 No. 2 interrupts the PC CPU, interrupt program I17 is executed. (I16 has higher priority.)

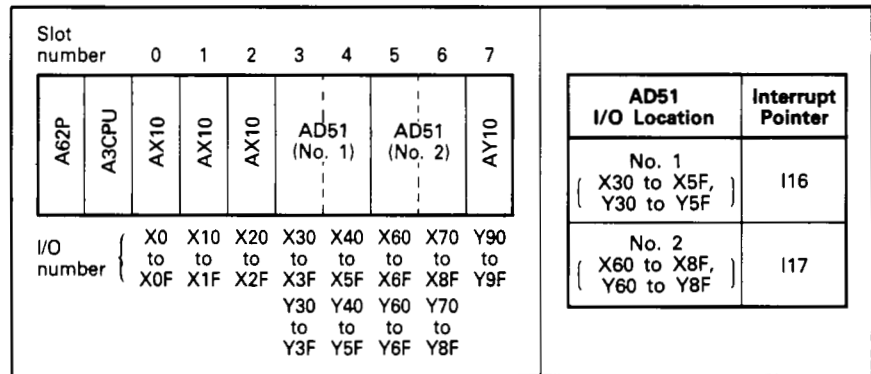


Fig. 7.7 System Configuration and Interrupt Pointers

7.11 Remote RUN/STOP of PC CPU

The PC CPU can be switched between RUN and STOP by the AD51 using the following system subroutines.

| Item | | System Subroutine | Processing |
|--------|-------------|-------------------|---------------------------------|
| PC CPU | Remote RUN | SKR | Requests remote RUN of PC CPU. |
| | Remote STOP | SKP | Requests remote STOP of PC CPU. |

(1) Precautions for remote RUN/STOP

- 1) Note that a "remote error" is flagged if a remote RUN (or STOP) signal is given to a PC CPU which has already received a remote STOP (or RUN) signal from a separate unit, e.g. AJ71C24-S3.
- 2) Remote RUN/STOP commands from the AD51 are valid as follows for different CPU key switch positions:

| | | PC CPU Key Switch Position | | | |
|-------------------|-------------|----------------------------|------|-------|----------|
| | | RUN | STOP | PAUSE | STEP-RUN |
| Command from AD51 | Remote RUN | RUN | STOP | PAUSE | STEP-RUN |
| | Remote STOP | STOP | STOP | STOP | STOP |

- 3) The clearing of data memories on receiving a remote run instruction depends on the states of special relays M9016 and M9017.

| Special Relay | | Data Memory State |
|---------------|--------|--|
| M9016 | M9017 | |
| OFF | OFF | CPU is run without clearing data memory. |
| OFF | ON | Data memory is cleared outside the latch range set in parameters. (Link image is not cleared.) |
| ON | ON/OFF | CPU is run after data memory is cleared. |

REMARKS

Always reset special relays M9016 and M9017 where data memory clearing is not required.

- 4) Resetting the PC CPU during remote RUN/STOP control (either with the keyswitch or by powering down and up) causes the remote signal to be removed and the PC CPU to revert to the mode detected by its key switch.

(2) BASIC program example

[Program example]

Program for remote RUN/STOP of PC CPU through key inputs

```
100 LOCATE 20, 10 .....Specifies cursor position.
110 PRINT "STOP PC CPU? Y/N" .....Displays message.
120 A=INKEY.....Waits for key input.
130 IF A#"Y" GOTO 120 .....Checks key input characters.
140 Z=CALL(0, $ 8036, $ FF, 60) .....Executes remote STOP.
145 IF Z#0 PRINT "ERROR",
      Z; GOTO 140.....Checks for errors in SKP execution.
150 LOCATE 20, 10 .....Specifies cursor position.
160 PRINT "REMOTE-RUN PC CPU?
      Y/N" .....Dislays message.
170 A=INKEY.....Waits for key input.
180 IF A#"Y" GOTO 170 .....Checks key input characters.
190 Z=CALL(0, $ 8033, $ FF, 60) .....Executes remote RUN.
200 IF Z#0 PRINT "ERROR",
      Z; GOTO 190.....Checks for errors in SKR execution.
210 GOTO 100
```

8. TROUBLESHOOTING

This section lists error messages and troubleshooting procedures.

8.1 Screen Error Messages

The following messages may be generated during operation of the AD51 with its programming console.

| Error Message | Display Screen | Description | Corrective Action |
|--------------------------|-------------------------------|--|---|
| CANNOT SET | Mode select menu | 1) Invalid number has been set. 2) "1" (MULTI TASK GO) or "6" (SYSTEM DATA TRANSFER) has been pressed before multitask setting, or there is an error in the multitask data. | 1) Correct the number. 2) Set or correct multitask data. |
| | BASIC program address setting | Invalid number has been set. | Correct the number. |
| | Date and time setting | Invalid value has been set. | Correct. |
| | GPP mode | 1) GPP mode selected without connecting the GPP/HGP/PHP to CH1. 2) Invalid number has been set. | Correct. |
| MEMORY PROTECT ERROR | Mode select menu | System data area is memory protected. | Set the memory protect switch to OFF. |
| DATA [] SET ERROR | BASIC program address setting | The data indicated on the menu by the number n is wrong. | Correct. |
| | Printer setting | | Correct. |
| ERROR | Multitask setting | Value above "ERROR" is wrong. | For ERROR displayed in the TYPE, START CONDITION, or INTERVAL, columns, correct the data on the screen. For other columns correct the data on the BASIC program address setting screen. |
| AD51 BUS ERROR | GPP mode | GPP/HGP/PHP inaccessible to AD51 buffer. | Usually causes by the PC CPU accessing the AD51 buffer memory too frequently or with too much data. STOP the PC CPU. |
| AD51 COMMUNICATION ERROR | | Communication error between AD51 and GPP/HGP/PHP. | Check cable connection and start up again. |
| AD51 WRITE ERROR | | Memory area is ROM or memory protected. | Select RAM area channel or reset memory protect. |
| ADDRESS ERROR | | Address is not in the allowed range. | Correct. |
| CANNOT USE KANA!! | | "Kana" (i.e. Japanese characters) in the system name. | Select appropriate character set and use alphanumerics for system name specification. |
| DISK FULL | | FD capacity exceeded. | Insert new FD. |
| FILE NAME ERROR | | Invalid file name for file directory or delete function. | Correct. |
| FLOPPY ERROR | | 1) No FD in accessed drive. 2) FD is write protected. 3) FD is defective. | 1) Insert FD. 2) Set FD write protect tab to OK. 3) Change FD. |
| FLOPPY WRITE PROTECT | | FD is write protected. | Set FD write protect tab to OK. |
| IDENTICAL NAME | | The same file name exists. | Change the file name. |
| MEMORY NOTHING | | Invalid area number has been specified. | Correct the area number. |
| NO FILE | | Specified file is not on FD. | Correct. |
| ROM ERASING ERROR | | ROM has not been erased. | Erase ROM data or use a new ROM. |

| Error Message | Display Screen | Description | Corrective Action |
|--|------------------------|---|--|
| ROM WRITE ERROR | GPP mode | 1) ROM is wrongly or not loaded. 2) ROM is defective. | 1) Check ROM. 2) Write several times. Try again, if data cannot be written, change ROM. |
| SIZE UNMATCH (ROM < FILE) | | ROM capacity is smaller than file capacity. | Select appropriate ROM. |
| SYSTEM NAME ERROR | | Invalid name has been specified. (The name includes non-alphanumeric character or blank or the first character is not a letter.) | Correct. |
| VERIFY ERROR | | Data unmatched. | Correct. |
| STACK ERROR! AD51 STOP! | Multitask execution | Stack has been used outside the set area. | In BASIC, a maximum of ten levels of GOSUB or FOR/NEXT instructions are allowed. |
| BTWF ERROR! AD51 STOP! | | Task scheduling RAM data has been changed. | Check whether the system memory has been accessed by the user program. |
| WAIT ERROR! AD51 STOP! | | There is a BASIC statement which cannot be translated by the interpreter. | Correct BASIC program. |
| AD51 STOP! TASK NO. | | STOP command executed. | Remove STOP command or change to END, GOTO, GOSUB, RETURN, ONGOTO or ONGOSUB command. |
| STOP COMMAND AD51 STOP TASK NO. | | BREAK command executed. | Remove BREAK command. |
| BREAK COMMAND AD51 STOP! TASK NO. | | BASIC program does not finish with END, GOTO, GOSUB, ONGOTO, ONGOSUB or RETURN command. | Correct. |
| TEXT END AD51 STOP! TASK NO. | BASIC programming mode | BASIC programming mode error detected in BASIC program. *1 | Correct. |
| WHAT? | | Program area insufficient. | Expand. |
| HOW? | | Program area is ROM or memory protected. | Alarm message *2 |
| SORRY | | | |
| ROM OR MEMORY PROTECT AREA! PLEASE DO NOT CORRECT PROGRAM | | | |

POINT

***1: "WHAT" and "HOW" are indicated when:**

- 1) An undefined command is used;
- 2) A command description format is wrong;
- 3) A line number is not specified on the left of the GOTO, GOSUB, ONGOTO, or ONGOSUB command; and
- 4) The RETURN command is used without the GOSUB or ONGOSUB command.

***2: When this message is indicated, never correct the program.**

Correction will corrupt the BASIC program memory area data. With this message displayed, only LIST, LLIST and BYE commands should be used.

To allow correction of a protected program, switch the memory protect off.

8.2 Error Code List

The occurrence of any error during AD51 operation will cause the appropriate error code to be displayed on the two digit annunciator.

Code definitions are as follows:

| Error Number | Error | | Description | Location | Corrective Action |
|--------------|--------------------------|-------|---|----------|--|
| 00 | Battery error | | Battery is not loaded. Battery voltage low. | — | Load battery. Change battery. |
| 10 | Multi task setting error | | Although multi task setting is wrong, multi task has been executed. | — | Re-set multi task. |
| 11 | BASIC program error | | Grammatical error in BASIC program. | Task 1 | Correct program. |
| 12 | | | | Task 2 | |
| 13 | | | | Task 3 | |
| 14 | | | | Task 4 | |
| 15 | | | | Task 5 | |
| 16 | | | | Task 6 | |
| 17 | | | | Task 7 | |
| 18 | | | | Task 8 | |
| 21 | STOP error | BASIC | STOP command has been executed during multi task execution. | Task 1 | Remote STOP command or change to END, GOTO, GOSUB, ONGOTO, ONGOSUB, or RETURN command. |
| 22 | | | | Task 2 | |
| 23 | | | | Task 3 | |
| 24 | | | | Task 4 | |
| 25 | | | | Task 5 | |
| 26 | | | | Task 6 | |
| 27 | | | | Task 7 | |
| 28 | | | | Task 8 | |
| 31 | BREAK error | BASIC | BREAK command has been executed during multi task execution. | Task 1 | Remove BREAK command. |
| 32 | | | | Task 2 | |
| 33 | | | | Task 3 | |
| 34 | | | | Task 4 | |
| 35 | | | | Task 5 | |
| 36 | | | | Task 6 | |
| 37 | | | | Task 7 | |
| 38 | | | | Task 8 | |
| 41 | Text end error | | BASIC program does not end with END, GOTO, GOSUB, ONGOTO, ONGOSUB, or RETURN command. | Task 1 | Correct program. |
| 42 | | | | Task 2 | |
| 43 | | | | Task 3 | |
| 44 | | | | Task 4 | |
| 45 | | | | Task 5 | |
| 46 | | | | Task 6 | |
| 47 | | | | Task 7 | |
| 48 | | | | Task 8 | |

| Error Number | Error | Description | Location | Corrective Action |
|--------------|------------------------------|--|-------------|---|
| 51 | ORST error | A task has been re-started before it has completed. | Task 1 | Correct task start condition. |
| 52 | | | Task 2 | |
| 53 | | | Task 3 | |
| 54 | | | Task 4 | |
| 55 | | | Task 5 | |
| 56 | | | Task 6 | |
| 57 | | | Task 7 | |
| 58 | | | Task 8 | |
| 60 | Stack error | Stack used is outside the system stack area. | — | In BASIC a maximum of ten levels of GOSUB or FOR/NEXT instructions are allowed. |
| 70 | Duplex WAIT error BTWF error | RAM contents for system's task schedule have been rewritten. | — | Check whether the system memory has been accessed by the user program. |
| 81 | Receive buffer full error | 511 bytes of received data in receive buffer. | RS-422 CH1 | Do not send more than 512 bytes at one time. |
| 82 | | | RS-422 CH2 | |
| 83 | | | RS-232C CH3 | |
| 84 | | | RS-232C CH4 | |
| 91 | Send buffer full error | 127 bytes of send data in send buffer. | RS-422 CH1 | Check cables. Empty the external equipment receive buffer. |
| 92 | | | RS-422 CH2 | |
| 93 | | | RS-232C CH3 | |
| 94 | | | RS-232C CH4 | |
| 99 | PC CPU error | 1) PC CPU has been reset during communication. 2) Time out error has occurred during PC accessing by system sub-routine. 3) PC CPU error detected by WDT and communication has stopped. <i>Note: Error code 99 is sometimes displayed after an instantaneous power failure.</i> | — | AD51 program execution not directly affected. |

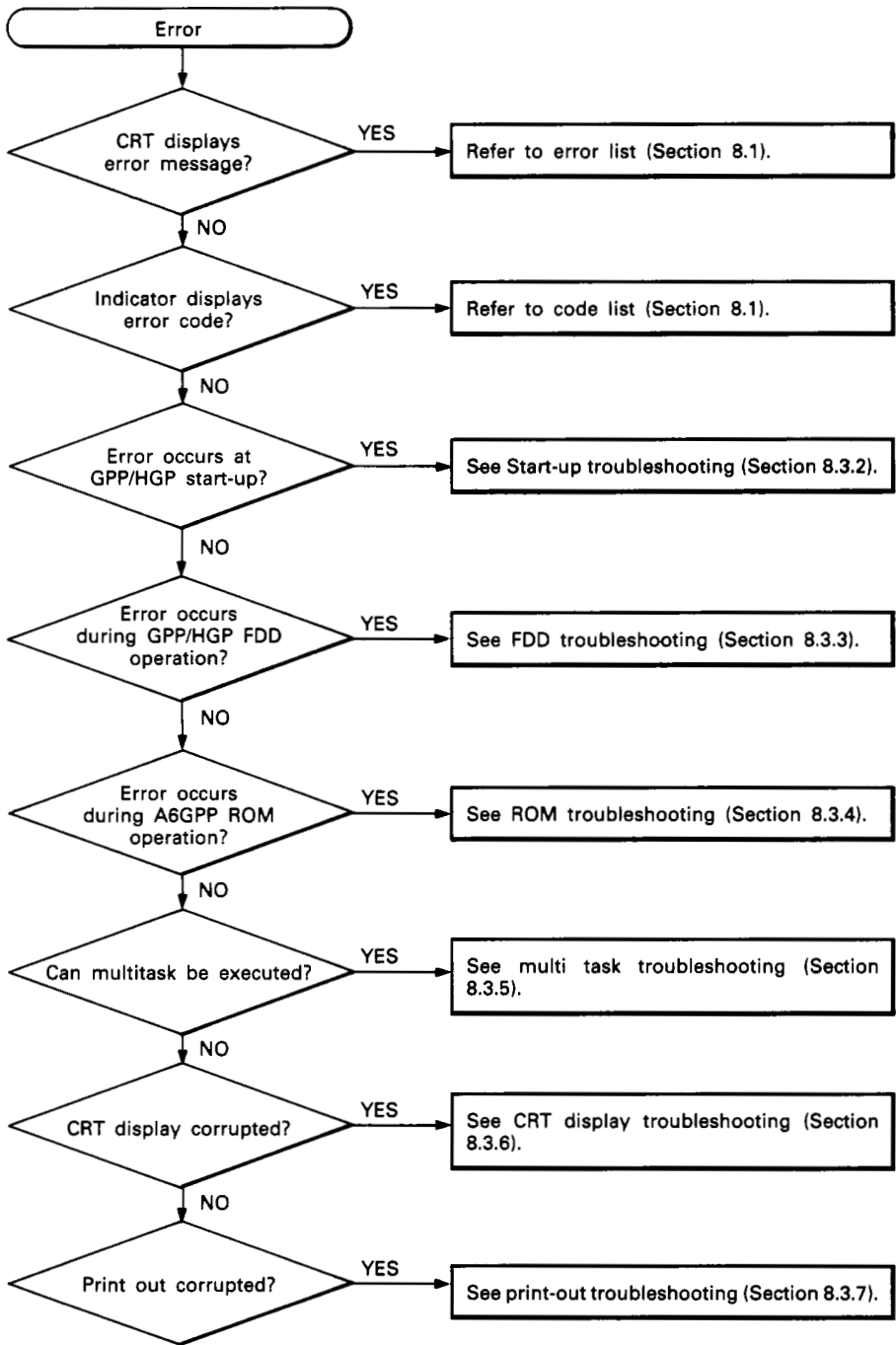
POINT

- (1) The AD51 continues operating in the event of "battery error", "ORST error", "receive buffer full error", "send buffer full error", or "PC CPU error".
- (2) The receive buffer capacity is 511 bytes per channel. Overflow data is ignored.
- (3) When data is sent from one task to the buffer memory and the buffer memory becomes full, the AD51 switches tasks. After that task has run the AD51 checks the buffer, if it is now vacant the original task is allowed to continue and if it is still full the AD51 will switch to a third task (where used). This checking and switching procedure will continue for 1 minute if the buffer remains full after which 1 byte of the excess data will be deleted.
The one minute cycle is repeated until there is no excess data left (The 1 minute cycle time may be changed using the SWB system subroutine.)
- (4) Possible causes of send buffer full error are as follows:
 - The DTR signal from the external equipment (Pin 6 of the RS-232C connector) is low.
 - X ON code is not received from the external equipment after X OFF has been received.

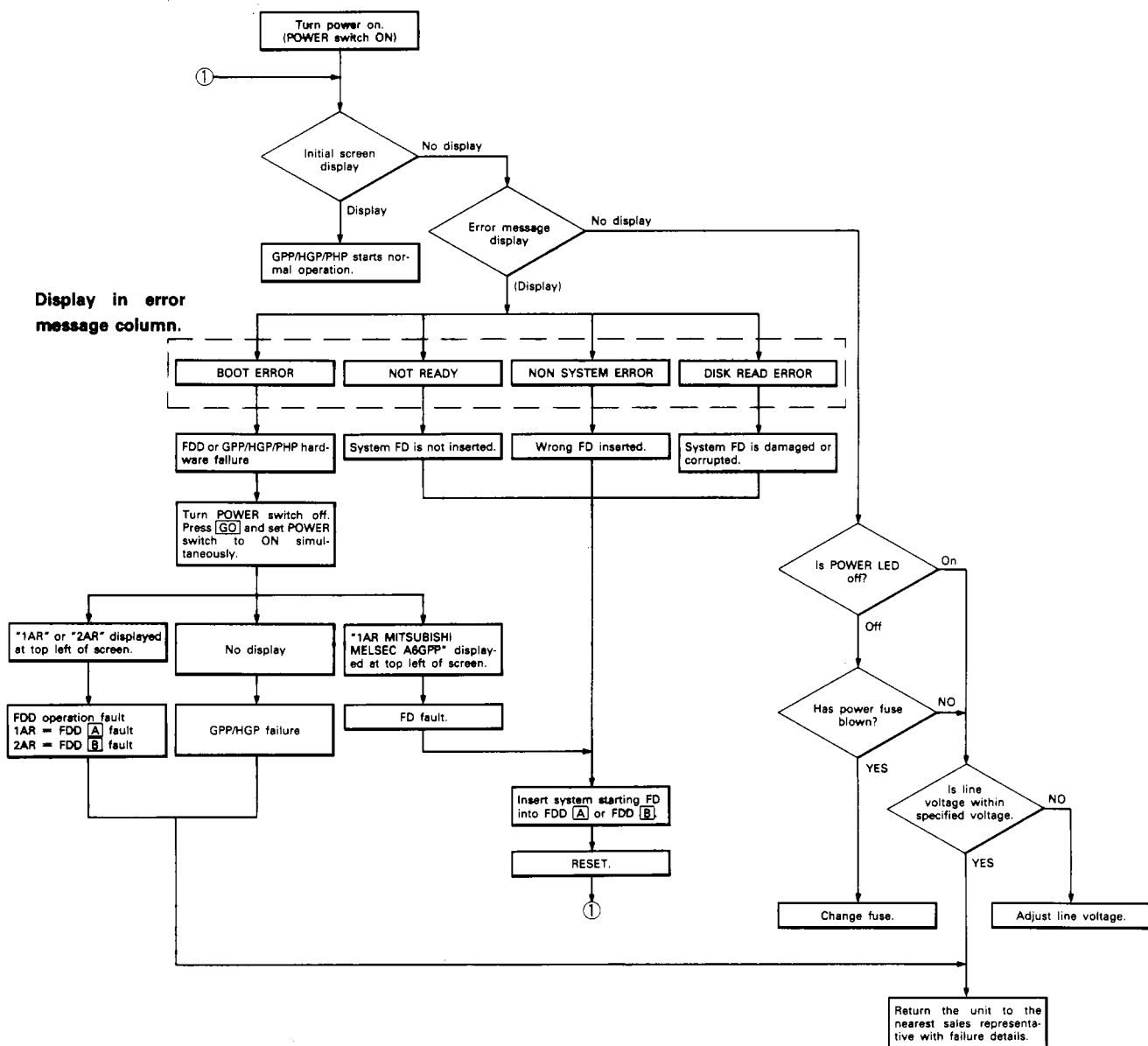
8.3 Troubleshooting

This section gives simple AD51 troubleshooting procedures. For PC CPU troubleshooting refer to the PC CPU User's Manual.

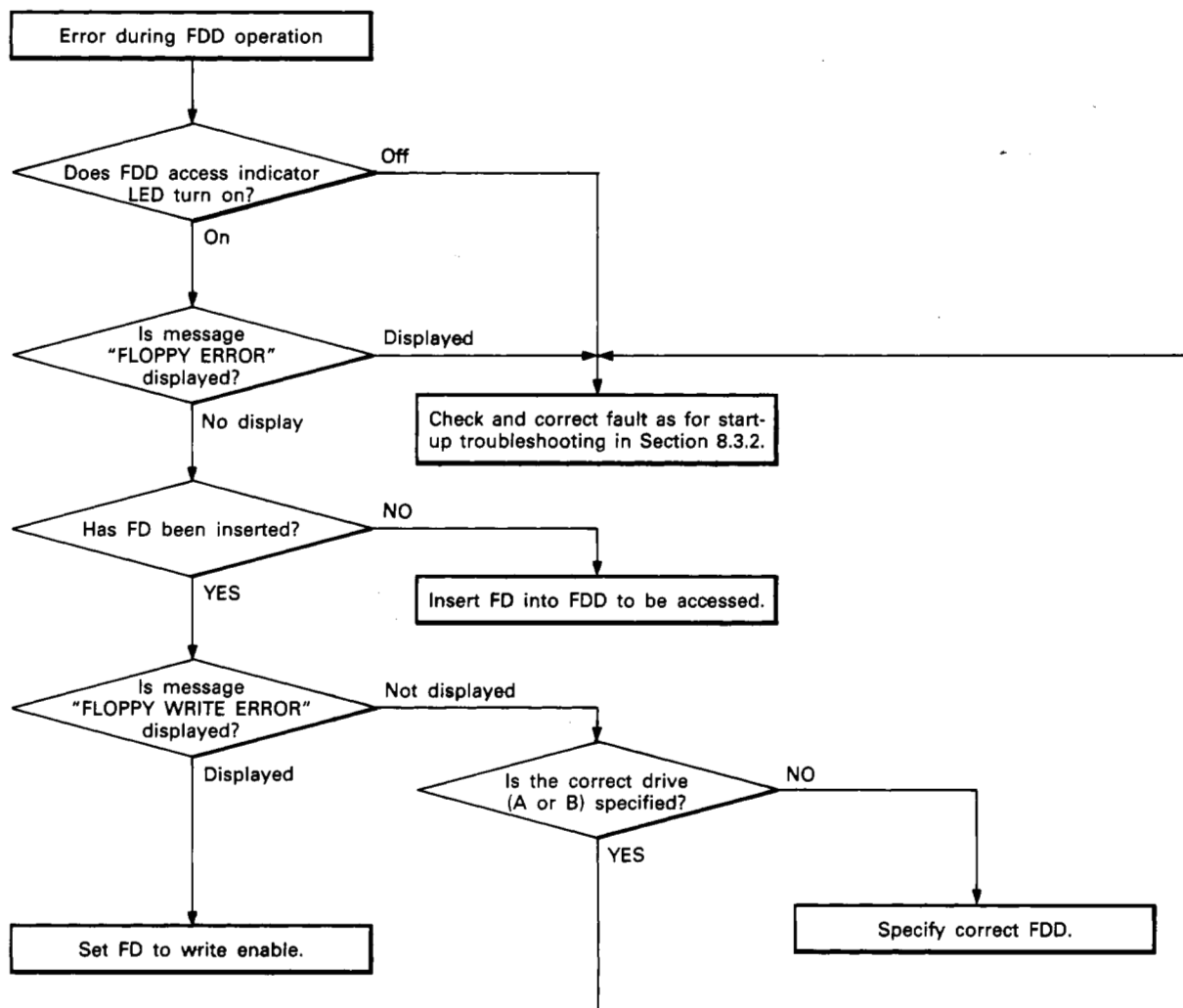
8.3.1 Troubleshooting flow chart



8.3.2 Start-up troubleshooting

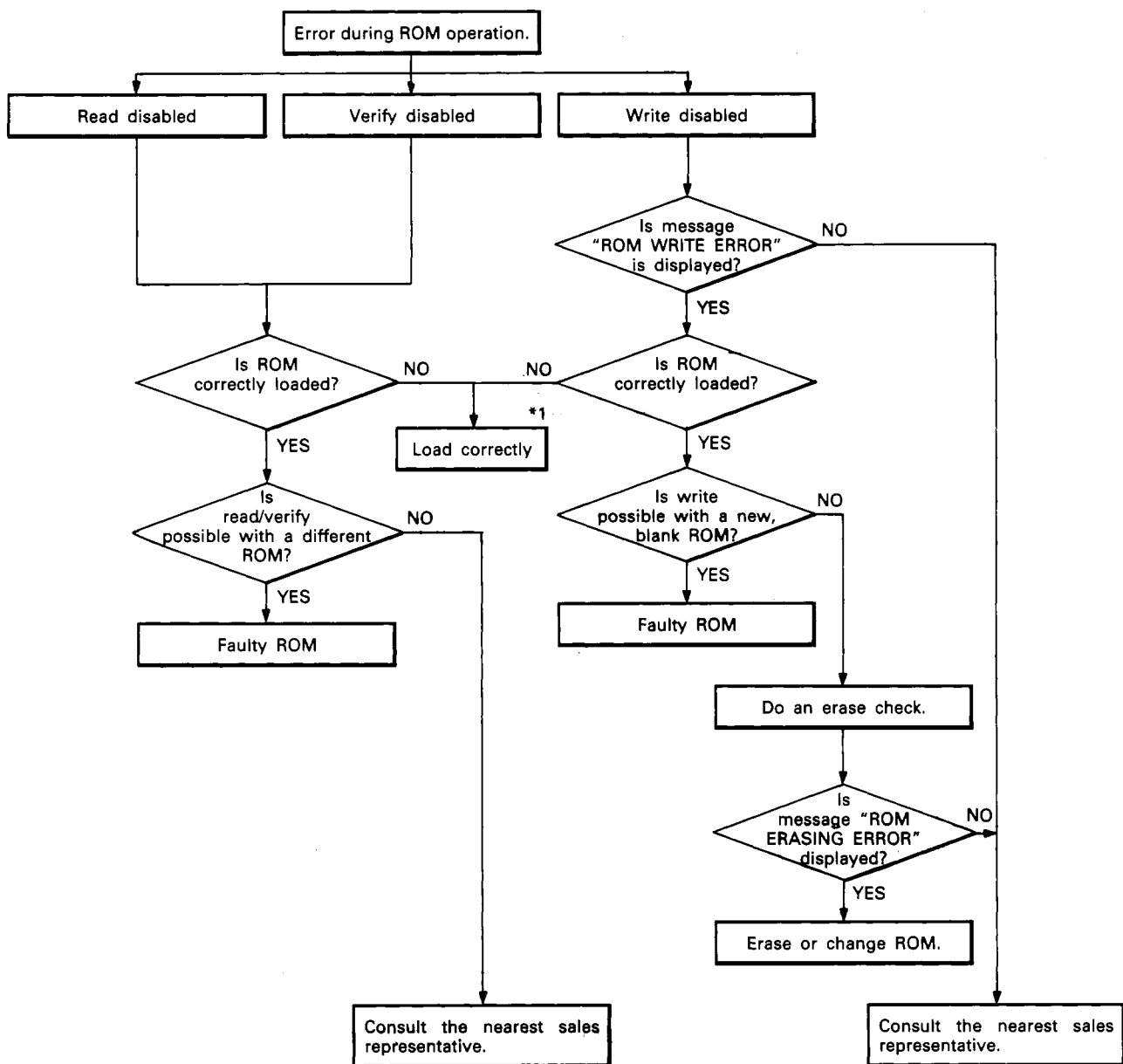


8.3.3 FDD troubleshooting

**POINT**

- (1) Note the correct direction for inserting the disk and never force it.
- (2) Formatting a disk clears all the data on it.

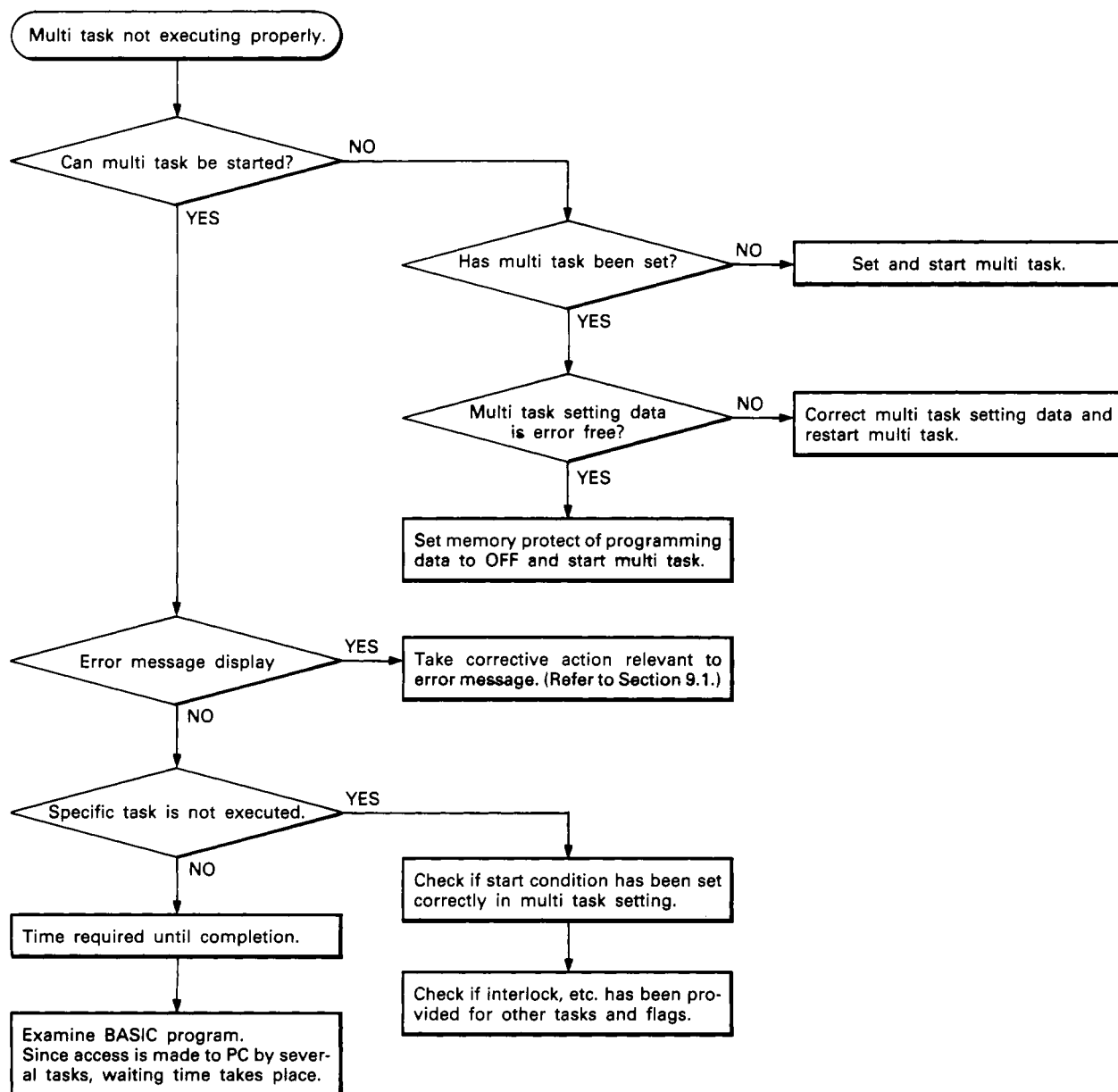
8.3.4 ROM troubleshooting



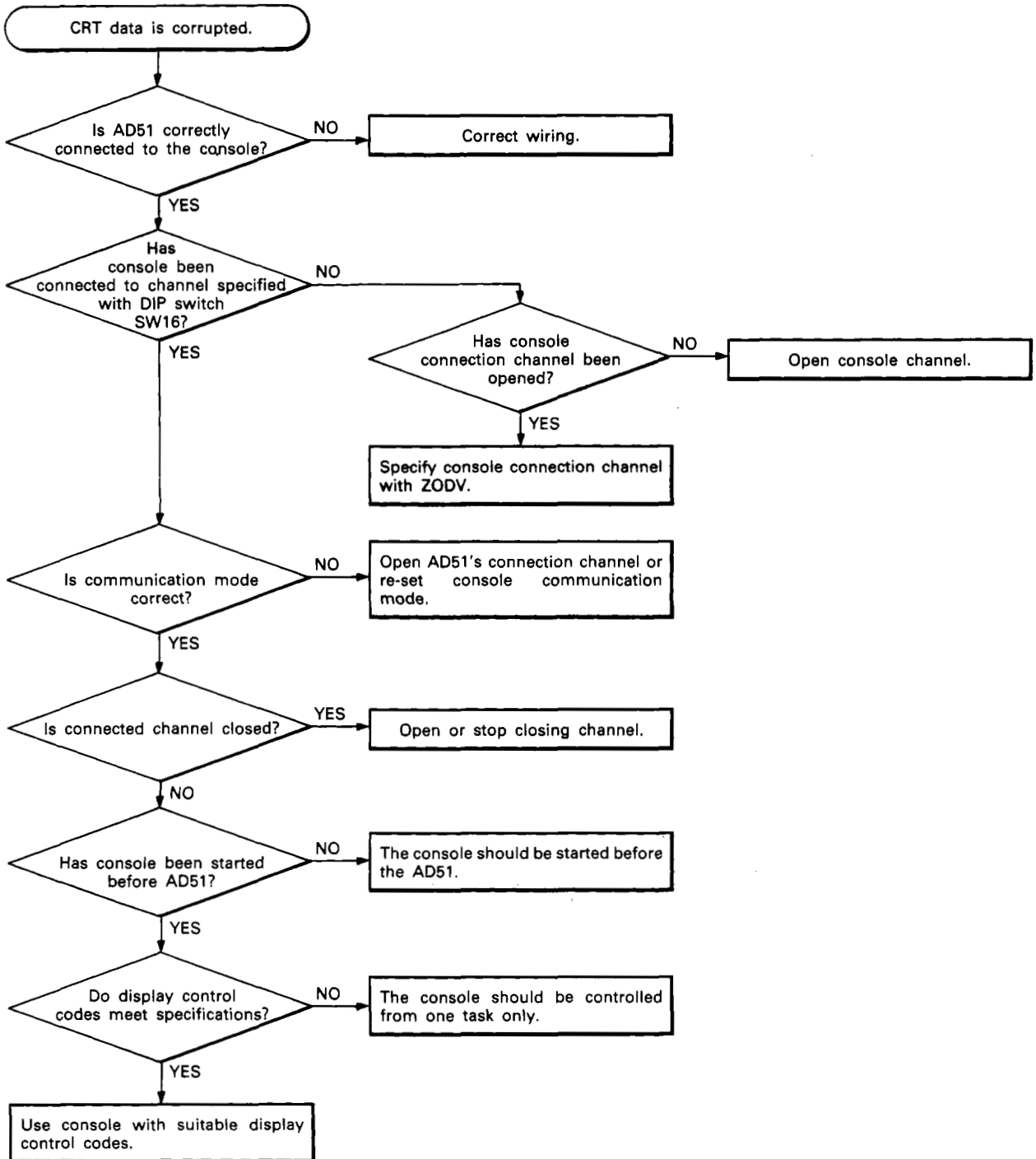
*1: Load correctly.

- 1) Is ROM in the right direction?
- 2) Is ROM securely loaded in socket?
- 3) Is ROM socket lever set correctly?

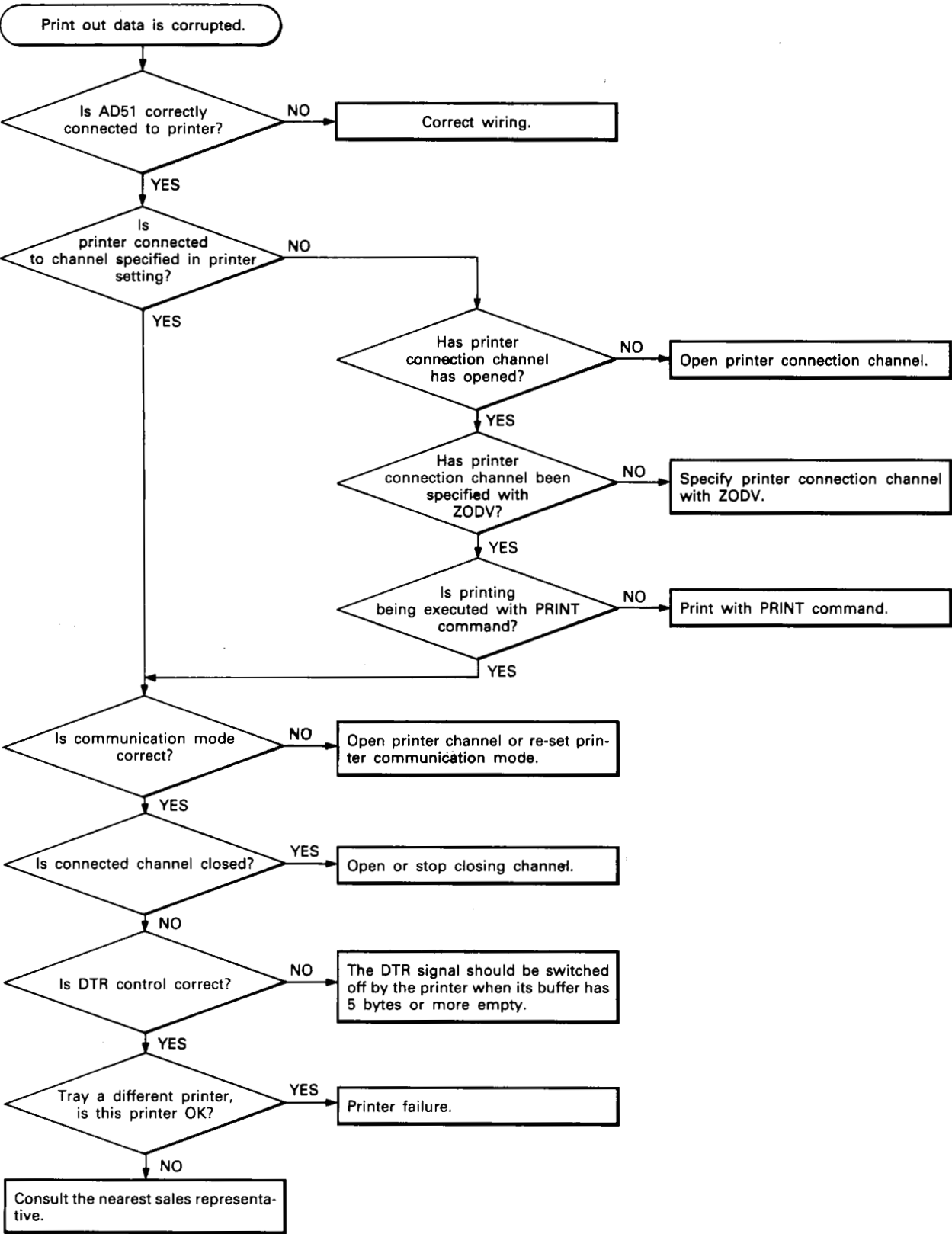
8.3.5 Multi task troubleshooting



8.3.6 CRT troubleshooting



8.3.7 Print-out troubleshooting



9. MAINTENANCE

9.1 Battery Life

When the battery voltage drops, the error indicator on the front of the AD51 displays "00". The error can also be read using the SIR system subroutine.

After this error message is displayed the battery has a further life of 65 days (1560 hours).

| | Guaranteed Value | Actual Operation Period (Av.) |
|-----------------------------|-----------------------|-------------------------------|
| Back-up by battery | 130 days (3120 hours) | 300 days (7200 hours) |
| Back-up after battery error | 65 days (1560 hours) | — |
| Back-up by capacitor | 11 minutes | 25 minutes |

Preventitive maintenance guide.

- (1) Change the battery after four years if the total battery back-up time during that period has been a maximum of 130 days.
- (2) For back-up periods exceeding a total of 130 days in four years, calculate the battery life as follows:

{Example}

Assume that the power is off for 14 hours 5 days a week, and all day for the remaining 2 days per week. Under these conditions, the power is off for:

$$\begin{aligned}
 14(\text{hours}) \times 5(\text{days}) &= 70 \text{ hours} \\
 24(\text{hours}) \times 2(\text{days}) &= 48 \text{ hours} \\
 &118 \text{ hours per week}
 \end{aligned}$$

The total battery life is 3120 hours which at 118 hours per week gives

$$\begin{aligned}
 \frac{3120}{118} &= 26.4 \text{ weeks} \\
 &= \text{Approx. 6 months.}
 \end{aligned}$$

Therefore,

it is necessary to change the battery every 6 months.

9.2 Battery Changing Procedure

Fig. 9.1 shows the battery changing procedure.

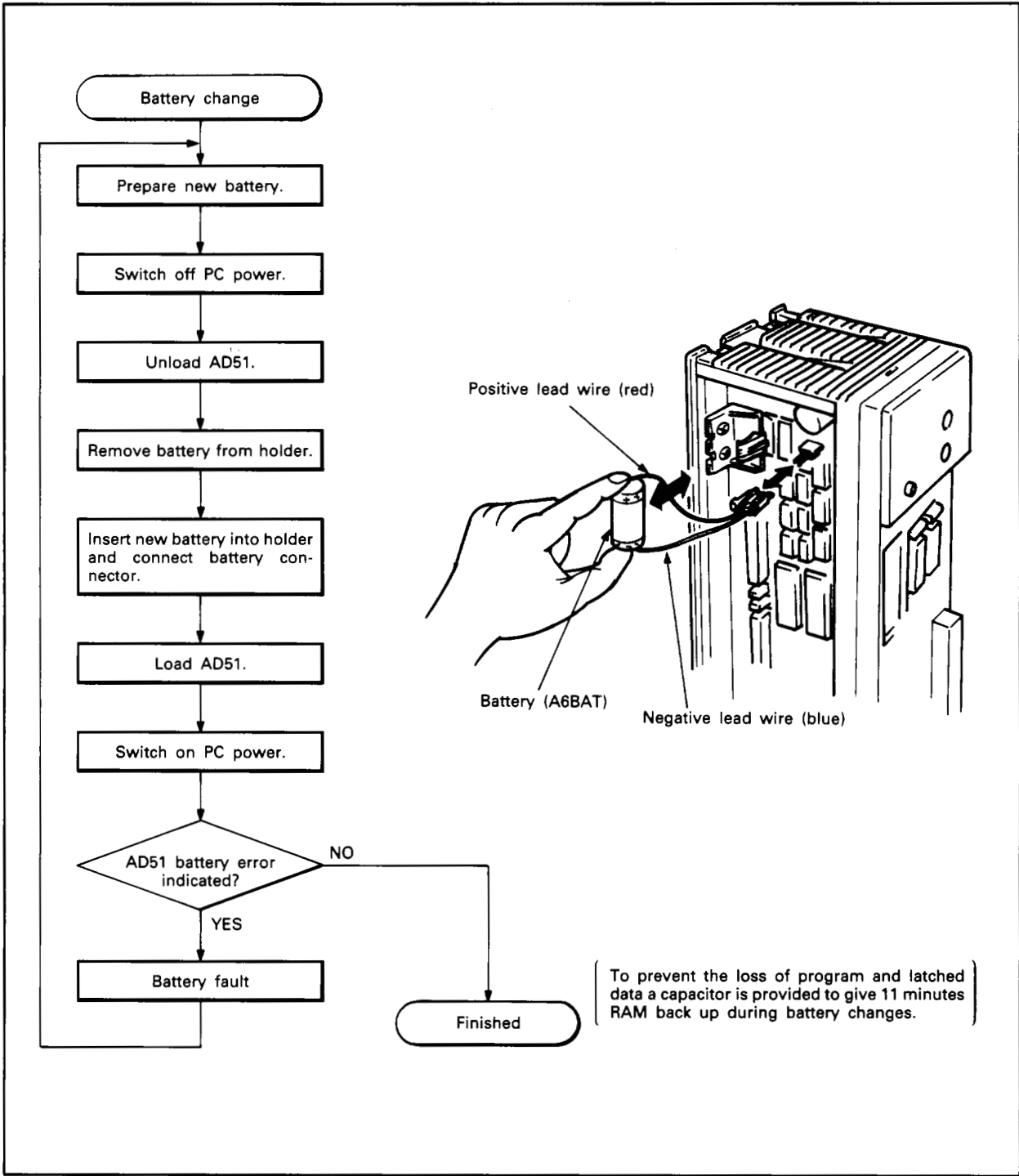


Fig. 9.1 Battery Changing Procedure

REMARKS

The battery is common to all the MELSEC-A series.

Battery storage life is 5 years. Total memory back-up guarantee period is 130 days.

Battery used is as follows.

Description : Lithium battery

Type and rating : Type A6BAT (3.6V with plug and leads)

Handling instructions

- (1) Do not short.
- (2) Do not disassemble.
- (3) Do not burn.
- (4) Do not heat.
- (5) Do not solder electrodes.
- (6) Do not measure voltage with an analog voltmeter.

APPENDICES

APPENDIX 1 Differences between AD51-S3 and AD51

The AD51-S3 differs from the AD51 in the following points:

- (1) The AD51-S3 allows communication with the other stations in MELSECNET.

The other stations can be accessed by specifying the MELSEC-NET PC No. at the set data head address of the system subroutine which used to access the PC CPU.

- (2) In addition to the AD51 system subroutines, the AD51-S3 has the following system subroutines:

| | System Subroutine | Function |
|----|-------------------|---|
| 1 | SFLTD | 32-bit integer → 32-bit floating point number |
| 2 | SFIXD | 32-bit floating point number → 32-bit integer |
| 3 | SAER | Reads data from extension file registers of PC CPU. |
| 4 | SAEW | Writes data to extension file registers of PC CPU. |
| 5 | SAET | Randomly writes data to extension file registers of PC CPU. |
| 6 | SAEM0 | Defines PC CPU extension file registers to be monitored. |
| 7 | SAEM1 | Monitors PC CPU extension file registers specified in monitor data entry. |
| 8 | SAMR | Reads microcomputer program from PC CPU. |
| 9 | SAMW | Writes microcomputer program to PC CPU. |
| 10 | SACR | Reads comments from PC CPU. |
| 11 | SACW | Writes comments to PC CPU. |
| 12 | SATR | Reads data from special function module buffer memory. |
| 13 | SATW | Writes data to special function module buffer memory. |

- (3) The AD51-S3 has system data transfer mode which allows system data to be transferred from 4F00_H-4FFF_H to 8000_H-80FF_H of channel 1.

This mode allows system data to be stored onto ROM.



APPENDIX 2 Special Function Module Buffer Memory Address Tables

The following tables list special function module buffer memory addresses specified by the AD51 using system subroutines SATR and SATW.

For full information on the buffer memory, see the corresponding module manual.

(1) A68AD analog-to-digital converter module

| Buffer Memory Assignment | Address Specified from AD51 | | Address for FROM/TO Instruction |
|------------------------------------|-----------------------------|-----------------|---------------------------------|
| | 8 lower bits | 8 higher bits | |
| Number of channels | 80 _H | 81 _H | 0 |
| Averaging processing specification | 82 _H | 83 _H | 1 |
| CH1 averaging time, count | 84 _H | 85 _H | 2 |
| CH2 averaging time, count | 86 _H | 87 _H | 3 |
| CH3 averaging time, count | 88 _H | 89 _H | 4 |
| CH4 averaging time, count | 8A _H | 8B _H | 5 |
| CH5 averaging time, count | 8C _H | 8D _H | 6 |
| CH6 averaging time, count | 8E _H | 8F _H | 7 |
| CH7 averaging time, count | 90 _H | 91 _H | 8 |
| CH8 averaging time, count | 92 _H | 93 _H | 9 |
| CH1 digital output value | 94 _H | 95 _H | 10 |
| CH2 digital output value | 96 _H | 97 _H | 11 |
| CH3 digital output value | 98 _H | 99 _H | 12 |
| CH4 digital output value | 9A _H | 9B _H | 13 |
| CH5 digital output value | 9C _H | 9D _H | 14 |
| CH6 digital output value | 9E _H | 9F _H | 15 |
| CH7 digital output value | A0 _H | A1 _H | 16 |
| CH8 digital output value | A2 _H | A3 _H | 17 |
| Write data error code | C4 _H | C5 _H | 34 |

(2) A62DA digital-to-analog converter module

| Buffer Memory Assignment | Address Specified from AD51 | | Address for FROM/TO Instruction |
|----------------------------------|-----------------------------|-----------------|---------------------------------|
| | 8 lower bits | 8 higher bits | |
| CH1 digital value | 10 _H | 11 _H | 0 |
| CH2 digital value | 12 _H | 13 _H | 1 |
| CH1 voltage set value check code | 14 _H | 15 _H | 2 |
| CH2 voltage set value check code | 16 _H | 17 _H | 3 |
| CH1 current set value check code | 18 _H | 19 _H | 4 |
| CH2 current set value check code | 1A _H | 1B _H | 5 |

(3) A84AD analog/digital converter module

| Buffer Memory Assignment | Address Specified from AD51 | | Address for FROM/TO Instruction |
|---|-----------------------------|-----------------|---------------------------------|
| | 8 lower bits | 8 higher bits | |
| Unused area | 10 _H | 11 _H | 0 |
| Averaging processing specification | 12 _H | 13 _H | 1 |
| CH1 averaging time, count specification | 14 _H | 15 _H | 2 |
| CH2 averaging time, count specification | 16 _H | 17 _H | 3 |
| CH3 averaging time, count specification | 18 _H | 19 _H | 4 |
| CH4 averaging time, count specification | 1A _H | 1B _H | 5 |
| Reserved area (must not be used) | — | — | — |
| CH1 digital I/O value | 24 _H | 25 _H | 10 |
| CH2 digital I/O value | 26 _H | 27 _H | 11 |
| CH3 digital I/O value | 28 _H | 29 _H | 12 |
| CH4 digital I/O value | 2A _H | 2B _H | 13 |
| CH1 internal setting mode flag | 2C _H | 2D _H | 14 |
| CH2 internal setting mode flag | 2E _H | 2F _H | 15 |
| CH3 internal setting mode flag | 30 _H | 31 _H | 16 |
| CH4 internal setting mode flag | 32 _H | 33 _H | 17 |
| CH1 temperature detection value | 34 _H | 35 _H | 18 |
| CH2 temperature detection value | 36 _H | 37 _H | 19 |
| CH3 temperature detection value | 38 _H | 39 _H | 20 |
| CH4 temperature detection value | 3A _H | 3B _H | 21 |
| CH1 set value check code | 3C _H | 3D _H | 22 |
| CH2 set value check code | 3E _H | 3F _H | 23 |
| CH3 set value check code | 40 _H | 41 _H | 24 |
| CH4 set value check code | 42 _H | 43 _H | 25 |
| Write data error code | 44 _H | 45 _H | 26 |
| Analog output enable signal enable/disable flag | 46 _H | 47 _H | 27 |
| CH1 module code | 48 _H | 49 _H | 28 |
| CH2 module code | 4A _H | 4B _H | 29 |
| CH3 module code | 4C _H | 4D _H | 30 |
| CH4 module code | 4E _H | 4F _H | 31 |
| CH1 temperature setting range (offset value) | 50 _H | 51 _H | 32 |
| CH1 temperature setting range (gain value) | 52 _H | 53 _H | 33 |
| CH2 temperature setting range (offset value) | 54 _H | 55 _H | 34 |
| CH2 temperature setting range (gain value) | 56 _H | 57 _H | 35 |
| CH3 temperature setting range (offset value) | 58 _H | 59 _H | 36 |
| CH3 temperature setting range (gain value) | 5A _H | 5B _H | 37 |
| CH4 temperature setting range (offset value) | 5C _H | 5D _H | 38 |
| CH4 temperature setting range (gain value) | 5E _H | 5F _H | 39 |

(4) AD61 high-speed counter module

| Buffer Memory Assignment | Address Specified from AD51 | | Address for FROM/TO Instruction | |
|-------------------------------|-----------------------------|-----------------|---------------------------------|-----|
| | Channel 1 | Channel 2 | CH1 | CH2 |
| Preset value write (lower) | 82 _H | C2 _H | 1 | 33 |
| Preset value write (middle) | 83 _H | C3 _H | | |
| Preset value write (upper) | 84 _H | C4 _H | 2 | 34 |
| | 85 _H | C5 _H | | |
| Mode register | 86 _H | C6 _H | 3 | 35 |
| | 87 _H | C7 _H | | |
| Present value read (lower) | 88 _H | C8 _H | 4 | 36 |
| Present value read (middle) | 89 _H | C9 _H | | |
| Present value read (upper) | 8A _H | CA _H | 5 | 37 |
| | 8B _H | CB _H | | |
| Set value read/write (lower) | 8C _H | CC _H | 6 | 38 |
| Set value read/write (middle) | 8D _H | CD _H | | |
| Set value read/write (upper) | 8E _H | CE _H | 7 | 39 |
| | 8F _H | CF _H | | |

(5) AD71(S1) positioning module

| Buffer Memory Assignment | | Address Specified from AD51 | Address for FROM/TO Instruction |
|-------------------------------|-------------------------|--|---------------------------------|
| X axis positioning start data | | 200 _H to 391 _H | 0 to 200 |
| Error reset | | 392 _H 393 _H | 201 |
| Y axis positioning start data | | 458 _H to 5E9 _H | 300 to 500 |
| Positioning data | X axis positioning data | 2040 _H to 235F _H | 3872 to 4271 |
| Positioning speed | | 2360 _H to 267F _H | 4272 to 4671 |
| Dwell time | | 2680 _H to 229F _H | 4672 to 5071 |
| Positioning address | | 29A0 _H to 2FDF _H | 5072 to 5871 |
| Positioning data | Y axis positioning data | 2FE0 _H to 32FF _H | 5872 to 6271 |
| Positioning speed | | 3300 _H to 361F _H | 6272 to 6671 |
| Dwell time | | 3620 _H to 393F _H | 6672 to 7071 |
| Positioning address | | 3940 _H to 3F7F _H | 7072 to 7871 |
| X axis parameter | | 3F80 _H to 3F9F _H | 7872 to 7887 |
| Y axis parameter | | 3FA8 _H to 3FC7 _H | 7892 to 7907 |
| X axis zeroing data | | 3FD0 _H to 3FDD _H | 7912 to 7917 |
| Y axis zeroing data | | 3FE4 _H to 3FF1 _H | 7922 to 7928 |



(6) AD72 positioning module

| Buffer Memory Assignment | Address Specified from AD51 | Address for FROM/TO Instruction |
|-------------------------------|--|---------------------------------|
| X axis positioning start data | 200 _H to 391 _H | 0 to 200 |
| Error reset | 392 _H 393 _H | 201 |
| Y axis positioning start data | 458 _H to 5E9 _H | 300 to 500 |
| Monitoring area | 6B0 _H to 6BF _H | 600 to 607 |
| X axis positioning data | 2040 _H to 2FDF _H | 3872 to 5871 |
| Y axis positioning data | 2FE0 _H to 3F7F _H | 5872 to 7871 |
| X axis parameter | 3F80 _H to 3F9F _H | 7872 to 7891 |
| Y axis parameter | 3FA8 _H to 3FC7 _H | 7892 to 7911 |
| X axis zeroing data | 3FD0 _H to 3FDD _H | 7912 to 7917 |
| Y axis zeroing data | 3FE4 _H to 3FF1 _H | 7922 to 7928 |

(7) AJ71C24-S3

| Address Specified from AD51 | Address for FROM/TO Instruction |
|--|---|
| 1000 _H to 11FF _H | 0 to FF _H |
| 1200 _H to 123F _H | 100 _H to (special application area) 11F _H |
| 1240 _H to 1FFF _H | 120 _H to 7FF _H |

APPENDIX 3 GPP/HGP Display Control Codes

| Function | Description | Code (ASCII) | BASIC Command |
|-------------------------|--|--|---------------|
| Line feed | Carriage return and line feed | CR, LF codes (0D _H , 0A _H) | — |
| Screen clear | All screen clear | FF code (0C _H) | CLS |
| XON | Enable transfer from external device. | DC1 code (11 _H) | — |
| XOFF | Disable transfer from external device. | DC3 code (13 _H) | — |
| Escape | Initiate escape sequence | ESC code (1B _H) | — |
| Back space | Cursor back one space | BS code (08 _H) | — |
| Cursor addressing | Set cursor position absolutely. | ESC + Y (59 _H) + line specification code (20 _H to 9F _H) + column specification code (20 _H to 9F _H) | LOCATE |
| Character qualification | Character highlight stop | ESC + O (4F _H) | ZNOR |
| | Character highlight | ESC + R (52 _H) | ZCRV |
| | Cursor ON | ESC + S (53 _H) | ZCON |
| | Cursor OFF | ESC + T (54 _H) | ZCOFF |
| Audible alarm | Bell | BEL code (07 _H) | — |

Display Control Code List

APPENDIX 4 GPP/HGP/PHP Key Codes and Character Codes

(1) GPP/HGP/PHP key codes

| | | | | | | | | | | | | | | | | | | | | | |
|----------------|----------------|----------------|----------------|----------------|---------|---------|----|---|---|---|---|-----|-------------|-----|---|---|---|---|---|---|-------|
| | | | | b ₈ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | | |
| | | | | b ₇ | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | | |
| | | | | b ₆ | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | |
| | | | | b ₅ | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | |
| b ₄ | b ₃ | b ₂ | b ₁ | Column Line | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F | ← HEX |
| 0 | 0 | 0 | 0 | 0 | | CTRL /P | SP | 0 | @ | P | | p | INS | | | — | | | | | |
| 0 | 0 | 0 | 1 | 1 | CTRL /A | CTRL /Q | ! | 1 | A | Q | a | q | ↑ | F1 | 。 | | | | | | |
| 0 | 0 | 1 | 0 | 2 | CTRL /B | CTRL /R | " | 2 | B | R | b | r | ↓ | F2 | [| | | | | | |
| 0 | 0 | 1 | 1 | 3 | BREAK | CTRL /S | # | 3 | C | S | c | s | → | F3 |] | | | | | | |
| 0 | 1 | 0 | 0 | 4 | CTRL /D | CTRL /T | \$ | 4 | D | T | d | t | ← | F4 | , | | | | | | |
| 0 | 1 | 0 | 1 | 5 | CTRL /E | CTRL /U | % | 5 | E | U | e | u | | F5 | . | | | | | | |
| 0 | 1 | 1 | 0 | 6 | CTRL /F | CTRL /V | & | 6 | F | V | f | v | | F6 | | | | | | | |
| 0 | 1 | 1 | 1 | 7 | CTRL /G | CTRL /W | ' | 7 | G | W | g | w | | F7 | | | | | | | |
| 1 | 0 | 0 | 0 | 8 | BS | CAN | (| 8 | H | X | h | x | HOME CLR | F8 | | | | | | | |
| 1 | 0 | 0 | 1 | 9 | HTAB | CTRL /Y |) | 9 | I | Y | i | y | | F9 | | | | | | | |
| 1 | 0 | 1 | 0 | A | LF | CTRL /Z | * | : | J | Z | j | z | | F10 | | | | | | | |
| 1 | 0 | 1 | 1 | B | CTRL /K | ESC | + | ; | K | [| k | { | | | | | | | | | |
| 1 | 1 | 0 | 0 | C | CTRL /L | CTRL /X | , | > | L | ¥ | l | | | | | | | | | | |
| 1 | 1 | 0 | 1 | D | CR | | — | = | M |] | m | } | | | | | | | | | |
| 1 | 1 | 1 | 0 | E | CTRL /N | | . | < | N | > | n | ~ | | | | | | | ° | | |
| 1 | 1 | 1 | 1 | F | CTRL /O | CTRL / | / | ? | O | _ | o | DEL | | | | | | | . | | |

↑
HEX

*For CTRL/ [], press [] and CTRL simultaneously.

| Key | Key Code |
|--------|-----------------------------|
| MELSAP | ESC + "(" (28H) + "M" (4DH) |
| GPP | ESC + "(" (28H) + "G" (37H) |

(2) GPP/HGP/PHP character codes

| | | | | | | | | | | | | | | | | | | | | | |
|----------------|----------------|----------------|----------------|----------------|----------------|---|---|----|---|---|---|---|---|---|---|---|---|---|---|---|-------|
| | | | | | b ₆ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | |
| | | | | | b ₇ | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | |
| | | | | | b ₈ | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | |
| | | | | | b ₅ | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | |
| b ₄ | b ₃ | b ₂ | b ₁ | Column Line | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F | ← HEX |
| 0 | 0 | 0 | 0 | 0 | | | | SP | 0 | @ | P | ' | p | ┐ | = | | - | | | α | × |
| 0 | 0 | 0 | 1 | 1 | | | | ! | 1 | A | Q | a | q | ┐ | ┐ | | | | | β | |
| 0 | 0 | 1 | 0 | 2 | | | | " | 2 | B | R | b | r | ↗ | ■ | | | | | ε | |
| 0 | 0 | 1 | 1 | 3 | | | | # | 3 | C | S | c | s | ↖ | ■ | | | | | μ | |
| 0 | 1 | 0 | 0 | 4 | | | | \$ | 4 | D | T | d | t | └ | ' | | | | | φ | |
| 0 | 1 | 0 | 1 | 5 | | | | % | 5 | E | U | e | u | ┘ | | | | | | θ | |
| 0 | 1 | 1 | 0 | 6 | | | | & | 6 | F | V | f | v | ┐ | ↘ | | | | | ω | |
| 0 | 1 | 1 | 1 | 7 | | | | ' | 7 | G | W | g | w | ┐ | └ | | | | | Ω | |
| 1 | 0 | 0 | 0 | 8 | | | | (| 8 | H | X | h | x | ┐ | | | | | | Σ | |
| 1 | 0 | 0 | 1 | 9 | | | |) | 9 | I | Y | i | y | ┐ | ┐ | | | | | π | |
| 1 | 0 | 1 | 0 | A | | | | * | : | J | Z | j | z | ↗ | | | | | | ↑ | |
| 1 | 0 | 1 | 1 | B | | | | + | ; | K | [| k | | ┐ | ┐ | | | | | ↓ | |
| 1 | 1 | 0 | 0 | C | | | | , | < | L | ¥ | l | | ┐ | ┐ | | | | | ▶ | |
| 1 | 1 | 0 | 1 | D | | | | - | = | M |] | m | | ┐ | ┐ | | | | | + | ■ |
| 1 | 1 | 1 | 0 | E | | | | . | > | N | < | n | ~ | ■ | → | | | | " | / | □ |
| 1 | 1 | 1 | 1 | F | | | | / | ? | O | - | o | ~ | K | ← | | | | . | \ | |
| | | | | | ↑ HEX | | | | | | | | | | | | | | | | |

(3) Selected character list

| Key Code | | English | German | Swedish | Japanese |
|----------|----|---------|--------|---------|----------|
| 1 | 5e | ^ | ^ | Ü | ^ |
| | 7e | ~ | ß | ü | ~ |
| 2 | 5c | \ | Ö | Ö | ¥ |
| | 7c | | ö | ö | |
| 3 | 40 | @ | § | É | @ |
| | 60 | . | . | é | . |
| 4 | 5b | [| Ä | Ä | [|
| | 7b | { | ä | ä | { |
| 5 | 5d |] | Ü | Å |] |
| | 7d | } | ü | å | } |
| 6 | 24 | \$ | \$ | ○ | \$ |
| 7 | 23 | £ | # | # | # |

Escape Sequence

| | Function | Description | Code |
|---|-------------------------|-----------------------------------|---|
| 1 | Screen clear | All screen erase | ESC + [+ 2 + J |
| 2 | Cursor addressing | Set cursor position. | ESC + [+ (line specification) + ; + (column specification) + H (*) |
| 3 | Character qualification | Character qualification OFF | ESC + [+ 0 + m |
| | | Character highlight | ESC + [+ 7 + m |
| 4 | Cursor home | Move the cursor to home position. | ESC + [+ H |

*: Line specification 1 to 24

Column specification 1 to 80

Example: To specify line 5 and column 10

ESC + [+ 5 (35_H) + ; + 1 (31_H) + 0 (30_H) + H

Line specification Column specification

Note: The LOCATE command counts the line and column, starting at 0. If "LOCATE 0, 0" is executed, code "ESC + [+ 1 + ; + 1 + H" is transmitted to the VT220.

APPENDIX 5 Storing the AD51E Memory Data into ROM Using the A6WU

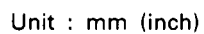
The AD51E internal memory and buffer memory data can be stored into the ROM with the A6WU P-ROM writer unit connected with the PC CPU.
For the operating procedure, see the A6WU Operating Manual.

The addresses must be set as follows when the AD51E data is written to the ROM using the A6WU.

| | | AD51E Addresses | Addresses Set by A6WU |
|------------------|-----|---|--|
| Programming data | | 4F81 _H to 4FD0 _H | 4F81 _H to 4FD0 _H |
| Common area | | 6000 _H to 67FF _H | 6000 _H to 67FF _H |
| Channel area | CH1 | 8000 _H to FFFF _H | 8000 _H to FFFF _H |
| | CH2 | 8000 _H to FFFF _H | 10000 _H to 17FFF _H |
| | CH3 | 8000 _H to DFFF _H | 18000 _H to 1DFFF _H |
| | CH4 | 8000 _H to DFFF _H | 20000 _H to 25FFF _H |
| Buffer memory | | 000 _H to BFF _H (0 to 3071) | 0 to 3071 |

REMARKS

Data may only be stored to the ROM if the AD51 is connected with the A1(E), A2(E), A3(E), A1N, A2N, A3N or A3HCPU.



IMPORTANT

The components on the printed circuit boards will be damaged by static electricity, so avoid handling them directly. If it is necessary to handle them take the following precautions.

- (1) Ground human body and work bench.**
- (2) Do not touch the conductive areas of the printed circuit board and its electrical parts with any non-grounded tools etc.**

Under no circumstances will Mitsubishi Electric be liable or responsible for any consequential damage that may arise as a result of the installation or use of this equipment.

All examples and diagrams shown in this manual are intended only as an aid to understanding the text, not to guarantee operation. Mitsubishi Electric will accept no responsibility for actual use of the product based on these illustrative examples.

Owing to the very great variety in possible applications of this equipment, you must satisfy yourself as to its suitability for your specific application.



mitsubishi electric corporation

HEAD OFFICE: MITSUBISHI DENKI BLDG MARUNOUCHI TOKYO 100 TELEX: J24532 CABLE MELCO TOKYO
NAGOYA WORKS : 1-14, YADA-MINAMI 5, HIGASHI-KU, NAGOYA, JAPAN

These products or technologies are subject to Japanese and/or COCOM
strategic restrictions and diversion contrary thereto is prohibited.

IB (NA) 66189-A (8901) MEE Printed in Japan

Specifications subject to change without notice.